CENA 10.000 zł
ISSN 0867-3918 INDEKS 377112

# 64 PLUS 4 & AMIGA

1/'92

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE



Razem
z legendarnymi bohaterami gier przygodowych
życzymy wszystkim Czytelnikom
**w Nowym 1992 Roku**
ciekawych i szczęśliwie kończących się
przygód w pracy z komputerem
i w życiu osobistym




REDAKCJA

## Drodzy Czytelnicy!

Bez wielkich jubileuszy rozpoczęliśmy kolejny rok wydawania naszego pisma. Kolorowa okładka i zwiększona objętość to następne kroki na drodze ku lepszej jakości.

Wiele osób sygnalizuje nam kłopoty z naszym - drukowanym wewnątrz numeru - blankietem wpłat. Okazuje się, że panie na poczcie nie bardzo wiedzą co zrobić z niektórymi odcinkami. W związku z tym wyjaśniamy, że jeden odcinek jest dla poczty, drugi dla wpłacającego , pozostałe dwa poczta powinna przesłać do naszego banku. Jeden z nich zostaje w banku, drugi przekazywany jest nam i na jego podstawie orientujemy się w treści zamówienia. Wszystkie odcinki są jednoznacznie opisane! W przypadku zwykłego blankietu wpłat (3-odcinkowego) zdarzało się często, że otrzymywaliśmy z banku tylko informację o wpłaconej kwocie - nie wiedzieliśmy więc kto i co zamawia. Było to przyczyną opóźnień w realizacji Waszych zleceń.

Przy okazji zwracamy uwagę na prawidłowe wypełnianie blankietów wpłat. Otrzymujemy sporo zamówień na których odnotowane jest tylko nazwisko i miasto - bez dokładnego adresu. W wielu przypadkach przyczyną opóźnień jest długi czas przepływu Waszych pieniędzy. Z porównania dat nadania i daty otrzymania zamówienia wynika, że „szły" one trzy tygodnie (ależ nasz kraj jest wielki)! Prosimy więc o odrobinę wyrozumienia - zapewniamy Was, że dokładamy wszelkich starań, aby jak najszybciej realizować wszystkie zamówienia.

REDAKCJA

Wadliwa dystrybucja „64 plus 4 & Amiga" przez przedsiębiorstwo RUCH jest przyczyną kłopotów, jakie mają czytelnicy chcący nabyć nasze pismo. Zdarza się, że otrzymujemy jako zwroty NIETKNIĘTE paczki zbiorcze. Tymczasem czytelnicy sygnalizują, że do wielu kiosków nie dociera ono wcale. Dlatego:

**zapraszamy wszystkich chętnych do prowadzenia kolportażu „64 plus 4 & Amiga" (kluby, studia i sklepy komputerowe, księgarnie, osoby indywidualne itd.) do współpracy!**

**Oferujemy korzystne warunki!**

Zainteresowanych prosimy o kontakt z działem dystrybucji pod adresem: Przedsiębiorstwo ABUK, 87-200 Wąbrzeźno, ul. 1 Maja 33.

Przedsiębiorstwo ABUK S-ka z o.o. oferuje państwu **szybką i tanią obsługę reklamową.** Ogłoszenia drobne od osób indywidualnych (do 10 słów) przyjmujemy bezpłatnie. Większe - 1000 zł za słowo. Reklamy ramkowe (minimalny format - 20 $cm^2$): $1cm^2$ ogłoszenia -**8000zł**, cała strona - **3,0 mln zł**; kolor - odpowiednio 100% drożej. Ogłoszenia przyjmujemy za pośrednictwem poczty (nasz adres - patrz stopka redakcyjna). Treść ogłoszenia z określeniem formatu reklamy (ewentualnie zamówieniem koloru) prosimy nadsyłać listem poleconym wraz z odcinkiem wpłaty. Wpłat prosimy dokonywać za pomocą przekazu pieniężnego na konto Przedsiębiorstwa ABUK, Bank Polska Kasa Opieki SA Oddział w Bydgoszczy, konto nr : 5.09011-400522.7-136-11-111.0. Dołączenie do zamówienia odcinka wpłaty przyspieszy zamieszczenie reklamy. Redakcja nie ponosi odpowiedzialności za treść i wiarygodność ogłoszeń.




miesięcznik nr 1(15)
styczeń 1992
cena 1 egz.: 10.000 zł.

64PLUS4

WYDAWCA: ABUK Spółka z o.o.
REDAGUJĄ: Waldemar Szczygieł (redaktor naczelny) z zespołem.
ADRES REDAKCJI: Redakcja „64 plus 4", 85-166 Bydgoszcz 43, skrytka pocztowa 64.
OKŁADKA: Piotr Bartz.
SKŁAD: ABUK
DRUK: W.Z.G. Wąbrzeźno, okładka: Z.P. POLRASTER, Bydgoszcz, .

## OD REDAKCJI

# W numerze :

Od redakcji .........3
Z daleka i z bliska ....4
Uczymy się programować ........5
Kupiłem C-64 i co dalej? ...........8
Doświadczenia z Voicetracker'em ....10
Programy ciekawe, zwariowane i takie sobie ........11
Assembler 6510 - lekcja 6 ...........12
Listy, listy ..........14
Public Domain Pack ..15
Reklama ...........17
Virus Expert II .......19
Kącik początkującego kodera ............20
ARP library - cz. 3 ...24
Gracz doskonały ....25
Kurs języka C .......26
Narysować kreskę czyli znowu małe co nieco o blitterze ...29

NOWINKI




# MAGAZYNY, MAGAZYNY...

*Ostatnio w Polsce coraz więcej posiadaczy komputerów C-64 zajmuje się programowaniem. Tym samym powstają coraz to nowe grupy, które zrzeszają ludzi zafascynowanych możliwościami tej maszyny. Niektóre z nich wydają swoje własne magazyny dyskowe informujące o działalności różnych grup na scenie komputerowej, o najnowszych demach, użytkach i grach, a także zawierające sporo artykułów na różne tematy wcale nie związane z komputerem.*
*Obecnie stale ukazują się w Polsce trzy magazyny: AXEL NEWS i PSC MAG redagowane w Gdyni oraz HIGHLIFE redagowany w Otwocku.*

- **AXEL NEWS** wydawany jest przez grupę AXEL z Gdyni. Ukazały się jak dotąd cztery numery tego magazynu. Wprawdzie początki nie były najlepsze jednak autorzy w każdym numerze stopniowo poprawiają program obsługujący magazyn, wprowadzając wiele znaczących zmian. Również teksty zawarte w magazynie, mimo ciągle jeszcze nie najwyższego poziomu, stają się coraz bardziej interesujące. Obecna szata graficzna magazynu w porównaniu z pierwszym jego numerem prezentuje się już całkiem nieźle.
Naszym zdaniem należałoby poprawić sposób obsługi magazynu, gdyż jest on niezbyt wygodny dla czytelnika. Poza tym wydaje się, że autorzy powinni bardziej przykładać się do jakości prezentowanych tekstów.
- **PSC MAG** wydawany jest przez gdyńską część grupy PARADOS. Magazyn ten ukazuje się już bardzo długo (ukazało się już aż jedenaście numerów). Początki, podobnie jak w przypadku innych magazynów, były bardzo słabe. Program wyświetlający teksty bardziej przypomina zwykłą notkę niż magazyn.
Po rozpadzie grupy PSC i przejęciu magazynu przez grupę PARADOS został napisany nowy program obsługi. Również poziom tekstów uległ znacznej poprawie, jednakże moim zdaniem nadal nie jest jeszcze w pełni zadowalający. PSC MAG jest magazynem robionym najbardziej pomysłowo. Uwagę zwracają samplowane „muzyczki" umieszczane przed magazynem, jak i dołączona gra, która uruchamia się po użyciu freeze lub naciśnięciu RESTORE.
- **HIGHLIFE** magazyn ten jest wydawany w Otwocku również przez grupę PARADOS. Swoją działalność rozpoczął on zaraz po założeniu tej grupy i jest jej oficjalnym miesięcznikiem. Od początku jego istnienia niewiele zmieniła się szata graficzna, chociaż w miarę upływu czasu wprowadzono kilka drobnych zmian, które jednak w większym stopniu nie wpływają na ogólny wygląd. Sama obsługa magazynu pozostawia również wiele do życzenia. Poziom tekstów jest całkiem niezły, chociaż zdarzają się numery, w których oprócz wiadomości nie było nic ciekawego. Myślę, że przydała by się nieco większa różnorodność publikowanych tekstów oraz kilka większych zmian w programie obsługi (na przykład dodanie pełnego spisu treści).

*W sumie bardzo trudno jest powiedzieć, który magazyn jest najlepszy, ponieważ każdy z nich posiada tyle samo wad co zalet. Najlepszym rozwiązaniem jest dokonanie oceny samemu!*
*Wszystkie najnowsze numery tych magazynów można otrzymać na większości giełd komputerowych, a także na kolejnych PD-pakach.*

C-16

W tej części naszego artykułu będziemy się zajmować zmiennymi funkcyjnymi. Rzadko kto wie, jak powinno się je właściwie stosować. Troszkę matematyki pozwoli lepiej zrozumieć ten bardzo użyteczny typ zmiennych.

# Uczymy się programować (cz. 4)

Przenosząc pamięć zmiennych na ekran, można zaobserwować, jak komputer administruje zmiennymi prostymi A, A%, A$ i FNA: koduje je w grupy siedmiobajtowe i układa w kolejności podawania w tabelę, której aktualny adres początku i końca zawarty jest pod wskaźnikiem 45/46 i 47/48 (patrz rys. 2 w numerze 10/91).

Wskaźnik, to para bajtów na stronie zerowej pamięci (pierwsze 256 bajtów pamięci komputera). Podstawiając do wzoru młodszy (L) i starszy (H) bajt oblicza się adres:

**AD=L+256xH**

Wzór Y(Z)=4xZ↑2 oznacza że, dla każdej liczby Z, dowolnie obranej zmiennej, można obliczyć, stosując reguły rachunku, po stronie prawej wartość liczbową zmiennej zależnej Y.

Podstawiając za Z wartość 3 otrzymuje się: Y(3)=3↑2=36.

Liczba w nawiasie oznacza dla jakiego Z obliczana jest wartość Y.

W Basic'u funkcja ta jest definiowana następująco:

```
10 DEF FNY(Z)=4*Z↑2
```

Lewa strona mówi że:

**DEFIniuję FUNkcję Y od Z.**

Dla Z=3 otrzymuje się przynależne Y(3) pisząc linię w Basic'u:

```
20 PRINT FNY(3)
```

Naturalnie nie można umieścić tej linii przed linią definiującą funkcję, gdyż pojawi się sygnał błędu „undefined function error" (brak definicji funkcji).

Co dzieje się w pamięci, jeśli funkcja zostanie zdefiniowana i przywołana?

Aby funkcja nie została zdefiniowana w trybie bezpośrednim, potrzebny jest mały program w Basic'u, który powinien być gotowy do pracy, zanim pamięć zmiennych zostanie umieszczona na ekranie.

```
10 def fny(z)=3*z↑2↑2↑2↑2↑2
20 z=4
30 ?fny(2/3)
```

Winii 20 widzimy, że nazwa zmiennej w nawiasie, może być równocześnie użyta do przechowywania wartości zwykłej zmiennej.

Umyślnie zdefiniowano funkcję w sposób tak skomplikowany, aby mieć do dyspozycji więcej czasu na obserwację.

Są dwie proste metody, za pomocą których można zaprogramować opóźnienie.

**Wskazówka 15**

Krótkie opóźnienie wytwarza rozkaz:

**a=5 ↑2↑2... ↑2↑2.**

Każde potęgowanie zabiera około 30 ms.

**Wskazówka 16**

Opóźnienie sekundowe otrzymuje się używając pętli: for i =1 to 1000: next.

Tysiąc przebiegów trwa około 1,3 sekundy.

W celu przeniesienia pamięci zmiennych na ekran, co praktykowaliśmy we wcześniejszych częściach, podaje się:

```
poke 45,0 :poke 46,12 :clr
```

Tabela zmiennych zaczyna się więc od adresu 0+12x256=3072 i jest to dokładnie lewy górny róg ekranu. Jeśli nie chce się obserwować żadnych łańcuchów, nie trzeba przenosić granicy pamięci (wskaźnik 55/56).

CLR zostało użyte, aby system mógł się dostosować do nowego podziału pamięci.

W celu łatwiejszego odszyfrowania ekranowego przedstawienie pamięci, przełączmy się w tryb pisowni małych liter i skasujmy ekran przyciskiem CLR/HOME. Następnie przesuńmy kursor dwie linie niżej i wystartujmy wspomniany wcześniej, program w Basic'u: RUN. Na ekranie u góry pojawiają się dwie grupy siedmiobajtowe.

Najpierw zmienne funcyjne, którą rozpoznaje się po pierwszym bajcie nazwy (w inwersji), a następnie zmienne zmiennoprzecinkowe Z o wartości 4, (nadana w linii 20). Jak pisaliśmy w 2 części naszego cyklu, zmienno-przecinkowe przedstawienie liczby 4 w kodzie ekranu, to małe c w inwersji z następującymi po nim czterema klamrami (zerami). Także bez linii 20 zostałoby zarezerwowane miejsce dla Z. Wszystkie bajty liczbowe miałyby w tym przypadku wartość zero.

W środku ekranu została wydrukowana wartość funkcji fny(2/3).

Bajt 3 i 4 (l=12 i p=16) zmiennych funkcyjnych zawiera wskaźnik początku definicji w pamięci programu, dokładniej mówiąc, miejsca za znakiem równości.

Adres, na który wskazuje wskaźnik to 12+16*256=4108.

Za pomocą

```
print peek (4108)
```

można zapytać o wartość ASCII tego miejsca pamięci. W odpowiedzi otrzymamy 51 i jak pokazuje tabela ASCII jest to cyfra „3", a więc pierwsze oznaczenie funkcji y. Dla kontroli powinno się także odczytać poprzedzające miejsce pamięci, które musi zawierać liczbę 178

Bajt 5 i 6 (i=9 i l=12) zawiera wskaźnik i również adres pierwszych bajtów liczbowych zmiennych z.

9+12x256=3081 to 10 pozycja na ekranie i dokładnie tam jest małe c - również w inwersji.

Bajt 7 zmiennej funkcyjnej zawiera pierwsze oznaczenie zdefiniowanej funkcji, a więc „3".

C-16

Funkcja zostanie także odszukana, kiedy kursor przesuniemy do górnej linii i z „3" zrobi się np. 8. Po zmianie przesuwa się z powrotem kursor na dół i podaje

**print fny (2/3)**

Zostanie wydrukowana ta sama liczba, jak po wykonaniu RUN. Uważnie obserwując zauważymy przy odczytywaniu fny zmieniające się wartości liczbowe Z w tabeli zmiennych. W celu dokładnego sprawdzenia, przesuwamy jeszcze raz kursor na ostatni rozkaz **Print** i naciskając przycisk **Return** uważnie obserwujemy bajt liczbowy. Podczas obliczeń zmienna zmienia wartość. Mimo relatywnie długiego czasu obliczeń trudno to rozpoznać. Trzeba jednak przyjąć, że jest to zmienno-przecinkowe przedstawienie „2/3".

Aby nie utracić pierwotnej wartości bajtu Z jest on zapamiętywany na stronie zerowej i po dokonaniu obliczeń jest znowu do dyspozycji.

Poniższe przykłady, powinny zachęcić do praktycznego używania zmiennych funkcyjnych. Zanim je wypróbujemy, należy komputer doprowadzić do stanu wyjściowego przyciskiem Reset.

**Wskazówka 17.**

Dla loteryjki należy wybrać sześć przypadkowych liczb od 1 do 49.

```
10   def   fny(z)=Int(49*rnd(1)+1)
20   for   I=1  to6:?fny(0),:next
```

W tej funkcji zmienna niezależna Z po prawej stronie nie występuje. Dlatego nie ma znaczenia, jaka liczba zostanie wpisana w nawiasie za fny, podobnie jak w przypadku fre(o).

Nazwa funkcji, tutaj y, musi być dokładnie podana przy jej przywoływaniu. Jeśli mamy więcej funkcji, to każda musi posiadać własną nazwę. Skąd komputer ma wiedzieć jaką regułę obliczeń przestrzegać?

Kolejny przykład: zaokrąglanie liczb na n miejscach po przecinku.

**Wskazówka 18.**

Jeśli na początku programu zdefiniuje się wzór zaokrąglania, to później w różnych miejscach programu, można się do niego odwoływać.

```
10   n=2
20   def   fnru(z)=Int(10↑n*z+.5)/10 ↑n
...
...
50   print   fnru(11.10*1.14)"dm"
...
70   n=3:a=5.7*1.234
80   print   fnru(a)"kg"
```

Jak widać w nawiasie mogą znajdować się także operatory rachunkowe (linia50) i dowolne zmienne (linia 80).

Definicje funkcji nie mogą być dłuższe niż jedna linia w Basic'u, a więc 80 bajtów, ale w przypadku bardziej skomplikowanych wzorów, mogą być one wzajemnie powiązane w celu skrócenia definicji.

```
10   REM   FUNKTIONS-DEMO
20   REM
30   :
40   DEF   FNA1(T)=90*EXP(-T/100)
50   DEF   FNA2(T)=FNA1(T)*(COS(T/10))
60   :
70   GRAPHIC1,1:DRAW319,100TO      0,100
80   FOR T=1 TO 319: DRAW TO T, 100-FNA2(T):NEXT
90   FOR T=1 TO 319 STEP 5:
     DRAW,T,100-FNA1(T):NEXT
100  GETKEY   A$:GRAPHIC0
```

**Listing 1: Demo z grafiką.**

Powyższy program pozwala na graficzne zobrazowanie drgań gasnących. Linia 70 opisuje oś czasu, a linia 90 opadającą krzywą wykładniczą, zdefiniowaną w linii 40. Powiązanie w tym przypadku polega na tym, że w definicji funkcji drugiej (linia 50), zawarta jest wartość funkcji pierwszej (linia 40).

Na zakończenie jeszcze jedna porada.

**Wskazówka 19.**

Pola zmiennych lub zmienne łańcuchowe (ARRAYS) wymagają własnego zakresu pamięci, ponieważ nie pasują do siedmiobajtowego schematu tabeli zmiennych. Adres początkowy obszaru zmiennych ARRAYS, który leży za tabelą zmiennych prostych, jest zapamiętany w parze bajtów 47/48 (patrz rys. - w poprzednim numerze).

Rysunek 5 zawiera przegląd sposobów przedstawienia wszystkich czterech typów zmiennych prostych w pamięci.

| Nr bajtu | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Typ | Nazwa | | | | | | |
| Zmienno-przecinkowe | <128 | <128 | Ekspo-nent | Mantysa | | | |
| Całkowite | ≥128 | ≥128 | high | low | 0 | 0 | 0 |
| Łańcuchowe | <128 | ≥128 | dług. | Wskaźnik tekstu low | high | 0 | 0 |
| Funkcyjne | ≥128 | <128 | Wskaźnik/Defin. low | high | Wskaźnik/Zmienne low | high | 1. Oznaczenie |

Rys. 5. Siedmiobajtowe przedstawienie zmiennych prostych.

W pierwszej części artykułu zwracaliśmy już uwagę na ogromne zużycie pamięci przez pola i na znaczenie ich wymiarowania (określania rozmiarów). W przypadku braku określenia wymiarów, komputer pozostawia dla każdego wymiaru miejsce na 11 zmiennych. Dla zmiennej zmienno-przecinkowej jest to pięć, dla liczby całkowitej dwa i dla string'ów trzy bajty. Dla wiernego przedstawienia trójwymiarowej tablicy zmienno-przecinkowej a (1,2,3) trzeba dokładnie 6666 bajtów. Jest to więcej niż połowa dostępnej pamięci w C-16. Przez określenie wymiarów: **dima(1,2,3)** obniża się zużycie do 131 bajtów.

Trzy liczby w nawiasie nazywa się indeksami.

Prosty przykład jednowymiarowego pola zmiennych całkowitych, powinien pokazać co się dzieje w pamięci po podaniu:

**dima%(5).**

Liczby całkowite jako zmienne pola są porządane, ponieważ wymagają o 3/5 pamięci mniej niż zmienne zmienno-przecinkowe.

Prosimy znowu „pamięć na ekran":

**poke45,0:poke46,12:clr.**

Po skasowaniu ekranu przesuwamy kursor dwie linie niżej i podajemy:

**dima%(5).**

Widzimy teraz linię składającą się z siedmiu oznaczeń i dwunastu klamr (zer). Komputer musi dla sześciu zmiennych pola a%(0), a%(1),...a%(5) zarezerwować miejsce w pamięci. Ponieważ zmiennym nie przydzielono jeszcze żadnej wartości liczbowej to komputer przypisuje im „wartości domyślne", czyli zero.

Pierwsza grupa oznaczeń przy polu jednowymiarowym składa się z siedmiu bajtów, a w przypadku dwuwymiarowego pola z dziewięciu, przedstawiających tzw. deskryptor pola, (patrz rys. 6.).

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... | ... | ... | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| Nazwa pola | | Długość | | Liczba wymiarów | ostatni wymiar | | ...... | | 1. wymiar | |
| | | high | low | | high | low | high | low | high | low |

Rys. 6

Bezpośrednio za dwoma bajtami nazwy, następuje określenie całkowitej długości pola.

W tym przypadku bajt low=19 (poz.3, rys.6, a bajt high=0 (poz.4, rys.6). Komputer musi więc od adresu 1 bajtu nazwy odliczyć 19 bajtów, aby dotrzeć do następnej zmiennej pola.

Ponieważ pole nie może przyjąć nowego wymiaru, długość całkowita jest więc wielkością stałą i może służyć jako wskaźnik połączenia lepiej niż adres, który każdorazowo zmieniałby się po wprowadzeniu nowej zmiennej prostej.

Pięć oznaczeń mówi ilu wymiarowe jest pole (a=1). Bajt 6 i 7 zawiera liczbę elementów, w naszym przykładzie: 6, (w tym wypadku bajt starszy stoi przed młodszym).

Wartości zmiennym pola można nadać wykorzystując pętlę **FOR-NEXT.**

Jest to wielka przewaga zmiennych pola.

Np. zmienne proste muszą być określone pojedyńczo:

```
forl=0to5:a%(l)=l:next
```

Po wprowadzeniu nowych zmiennych i, które są porządkowane przed polem, cała pamięć ARRAYS jest przesunięta na prawo o siedem miejsc. Za deskryptorem pola są zarezerwowane miejsca, które są teraz wypełniane liczbami całkowitymi.

Indeksami zmiennej pola mogą być wzory matematyczne. Komputer oblicza ich wartość i „odcina" miejsca po przecinku. Indeks nie może być jednak ujemny, ponieważ pojawia się „illegal quantity".

Dotychczas pod uwagę były brane tylko tablice jednowymiarowe. Dla każdego dalszego wymiaru deskryptor pola rośnie o dwa bajty, które podają ile elementów ten wymiar zawiera.

Ta nowa para bajtów jest „wciśnięta" za 5 bajtem, tak że wymiar znajduje się jako ostatni w nawiasie i rozpoczyna się dalej od szóstej pozycji deskryptora pola. Najlepiej pokaże to komputer po podaniu:

```
dima(1,2,3).
```

Trójwymiarowa tablica zmienno-przecinkowa ma 2x3x4=24 elementów i każdy element musi mieć 5 bajtów. Daje to razem z 11 pozycjami nagłówka 131 bajtów. Na ekranie ta liczba oznaczona jest przez małe c w inwersji (3+128).

Jak użyć zerowy element tablicy, pokazuje wskazówka.

**Wskazówka 20.**

Miesięczne wydatki jednego roku powinny być zebrane w pole a(i), a wartość średnia powinna być zapamiętana w a(0).

```
10dima(12):a(0)=0:forl=1to12:inputa(l)
:a(0)=a(0)+a(l)       :next:a(0)=a(0)/12:?a(0)
```

Jeżeli pole zmiennych zajmujące duży obszar pamięci nie jest już w programie potrzebne, można je skasować.

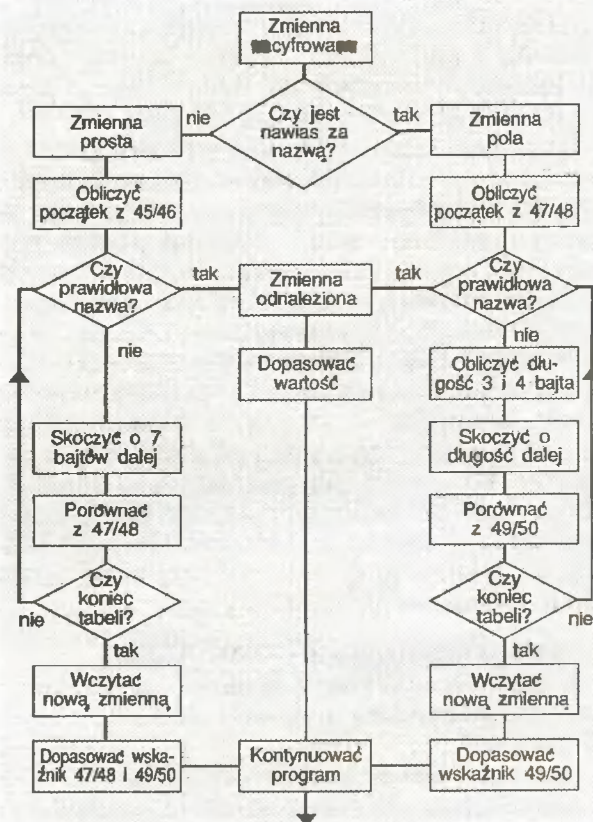
Administrowanie zmienną przysparza komputerowi wiele pracy. Rysunek 7 pokazuje konieczne kroki, które system musi wykonać, gdy w programie pojawia się zmienna.

**Wskazówka 21.**

Kasowanie pamięci ARRAYS bez naruszenia zmiennych prostych: dowiadujemy się jaki jest adres początku pamięci ARRAYS z pary bajtów 47/48 i wpisujemy go do pary bajtów 49/50.

Uzyskuje się to elegancko, tylko za pomocą dwóch rozkazów:

```
poke50,peek(48):poke49,peek(47)
```




Rys.7. Obróbka zmiennej.

Opracowano na podstawie RUN, nr 5/87

C-64

W końcu mam własny komputer. I co ważniejsze jest nim nasz upragniony COMMODORE 64. Po otwarciu pudełka nasuwa się jednak pytanie: co dalej? W tym atykule postaramy się odpowiedzieć na to pytanie.

# Kupiłem C-64 i co dalej?

**1. Jak to właściwie podłączyć?**

W pudełku oprócz samego komputera znajduje się dość długi kabel służący do podłączenia maszyny do telewizora oraz zasilacz. Jeżeli komputer chcemy używać z monitorem monochromatycznym (NEPTUN itp.) należy nabyć kabel mający z obu stron wtyczki typu DIN (mono albo stereo). Jeżeli posiadamy firmowy monitor COMMODORE to należy skorzystać z przewodu dołączonego do monitora. Podłączając nasz komputer do zwykłego telewizora używamy kabelek znajdujący się w pudełku z komputerem. Dźwięk i kolorowy obraz uzyskamy tylko na odbiorniku kolorowym pracującym w systemie PAL (C-64 korzysta z 36 kanału UHF).

Do naszego komputera można oczywiście podłączyć pamięci masowe, takie jak magnetofon, stacja dysków 5.25" lub 3.5" lub nawet dysk twardy. Magnetofon ma bardzo specyficzną wąską i wydłużoną wtyczkę, więc znalezienie odpowiedniego gniazda nie sprawia żadnych trudności, natomiast kabel od stacji dysków należy podłączyć do okrągłego gniazda opisanego SERIAL.

Gniazdo to służy również do podłączenia wielu innych urządzeń zewnętrznych (na przykład drukarki). Z boku komputera oprócz gniazda zasilacza znajdują się jeszcze wejścia służące do podłączenia dwóch joystick'ów. Po podłączeniu posiadanych urządzeń do naszej „maszyny" możemy już włączyć telewizor (lub monitor), stację dysków i na końcu sam komputer. Ukazuje się informacja o zainstalowanej wersji języka BASIC i... możemy przystąpić do pracy.

**2. Obsługa magnetofonu COMMODORE.**

Podstawową wadą pracy z magnetofonem jest bardzo długi czas wczytywania programów dochodzący nawet do 20 minut.

Do wczytywania programów służy instrukcja:

```
LOAD "nazwa",1
```

gdzie 'nazwa' jest tytułem programu, który chcemy wgrać do pamięci, a jedynka informuje komputer, że korzystamy z magnetofonu.

Jeżeli chcemy wgrać pierwszy program znajdujący się na taśmie, to wystarczy napisać tylko LOAD.

Do nagrania programu na taśmę służy instrukcja:

```
SAVE "nazwa",1
```

o parametrach analogicznych do LOAD.

Ze względu na długi czas standardowego wczytywania programów został opracowany dla COMMODORE specjalny system turbo, który przyspiesza transmisję danych około dziesięciokrotnie. Z tego sposobu zapisu korzystają prawie wszyscy użytkownicy C-64, tak więc większość programów dostępnych w Polsce na C-64 jest nagrana właśnie w tym systemie.

Aby wgrać program tak zapisany, wcześniej należy załadować przy pomocy instrukcji LOAD specjalny program umożliwiający korzystanie z turbo, który - po uruchomieniu wgrywanej w systemie turbo gry bądź innego rodzaju programu - ulega zniszczeniu. Tak więc najwygodniejszym rozwiązaniem jest umieszczenie programu turbo na początku krótkiej taśmy i korzystanie z niej przy każdorazowym załadowywaniu systemu.

W Polsce jest obecnie bardzo wiele programów służących do przyspieszenia transmisji danych z magnetofonu, różnią się one jednak jedynie nazwą i wyglądem zewnętrznym, gdyż sposób czytania danych jest zawsze taki sam. Godnym polecenia jest program TURBO ROM, który jest jednym z najlepszych tego rodzaju systemów. Wszystkie jego komendy rozpoczynają się poziomą strzałką ←, po której następuje litera. I tak na przykład litera L służy do wgrania programu, a S do nagrania go na taśmę.

Niektórym może wydawać się nieco męcząca konieczność ciągłego wgrywania turbo, dlatego też istnieją specjalne karty zawierające w stałej pamięci zapisany system turbo oraz dodatkowo wiele innych użytecznych funkcji. Najlepszą chyba dla użytkowników magnetofonu jest karta o nazwie „X". Oprócz dwóch rodzajów systemu turbo zawiera ona bardzo przydatny korektor głowicy magnetofonu, możliwość przedefiniowania klawiszy funkcyjnych, monitor języka maszynowego oraz prosty program kopiujący. Poza tym jest bardzo prosta w obsłudze.

**3. Obsługa stacji dysków.**

Współpraca COMMODORE ze stacją dysków nie należy do najszybszych. Często na wgranie programu dyskowego trzeba czekać ponad pięć minut co jest bardzo denerwujące.

Najważniejszą rzeczą jest wczytanie katalogu dyskietki - komenda:

```
LOAD "$",8:LIST
```

Katalog dyskietki wczytywany jest jako program w języku BASIC i można go przeczytać przy pomocy komendy LIST. Niestety wczytanie katalogu niszczy aktualnie znajdujący się w pamięci program.

Do wczytania programu służy komenda:

```
LOAD "nazwa",8,1
```

gdzie 'nazwa' jest tytułem programu, ósemka informuje, że korzystamy ze stacji dysków, a jedynka mówi, że program ma być wczytany w obszar pamięci, z którego był poprzednio nagrywany na dysk. Jeżeli jedynka zostanie pominięta dane będą załadowane od adresu początkowego programu w BASIC'u. Tytuł programu nigdy nie może zostać pominięty.

Do nagrania programu na dysk służy komenda:

```
SAVE "nazwa",8
```

i jej parametry są podobne do LOAD.

Stacji dysków można również wydawać komendy dyskowe nakazujące sformatowanie dyskietki, skasowanie pliku, czy zmianę jego nazwy, jednak wykonywanie tych czynności jest nieco bardziej skomplikowane. Na początku najlepiej korzystać z takich prgoramów, jak HACKEM COPY, które umożliwiają wykonanie tych operacji w bardzo prosty sposób.

Oczywiście podobnie jak do magnetofonu, tak i do stacji dysków opracowano bardzo wiele różnych systemów przyspieszających (od 5 do 20 razy). Jednakże w przypadku gier dyskowych systemy te nie sprawdzają się, gdyż gry najczęściej korzystają z własnych procedur ładowania.

Również dla użytkowników stacji dysków zostały opracowane różne rodzaje kart ułatwiających wykonywanie podstawowych operacji na dysku. Najbardziej popularne z nich to FINAL II, III oraz ACTION. Najlepszą dla początkujących użytkowników C-64 jest FINAL III. Cartridge ten posiada bardzo wygodną obsługę opartą na okienkach i dobrze rozbudowany interpreter języka BASIC.

Wkrótce postaram się omówić wszystkie najpopularniejsze karty rozszerzające dostępne na polskim rynku, a także napisać więcej o obsłudze stacji dysków z poziomu języka BASIC.

**Jarri**

# OGŁOSZENIA

KOMPUTEROWA FIRMA USŁUGOWA „TREND"
COMMODORE AMIGA 500 - 3000
LITERATURA W J. POLSKIM(!) I OPROGRAMOWANIE.
Informacja: dyskietka lub koperta + znaczek. Kontakt: Rafał Wierzbicki, ul. Budziszyńska 112/28, 54-436 Wrocław.

---

AMIGA 500/plus/2000 najlepsze gry i programy użytkowe, super nowości - sprzedaż wysyłkowa pocztą.
Ekspresowe terminy, katalogi gratis.
SOFTSTUDIO, ul. Tysiąclecia 54/6, 31-610 Kraków, tel. (012) 485150.

---

Sprzedam drukarkę CP-80X do C-64 z papierem i taśmami (1.4mln). Z. Żółtaszek, ul. Zapolskiej 61/9, 43-135 Tychy.

---

Wymienię programy na Amigę. Grzegorz TKACZYK, Oś. II BL 8 m.22, 16-400 Suwałki.

---

Sprzedam tanio sampler do A500. P. Borkowski, Z. Augusta 18/85, 76-200 Słupsk.

---

Przeróbka programów na Amigę 500/plus. Radosław Cymer, ul. Szafera 1/42, 92-306 Łódź.

---

Sprzedam przewód 5m do C-64 do podłączenia monitora AV COLOR. Kupię DATASSETTE 1530. J. Ziółkowski, ul. 26 Kwietnia 22/13, 72-009 Police, skr. 70.

---

C-64 KLUB. INFORMACJA: koperta + znaczek. 44-217 Rybnik, ul. K.B. Kominka 37b/33.

---

Amigowcu! Masz problem! Koperta + 2 znaczki. Leszek Gieniec, ul. Lwowska 136/1, 33-300 N. Sącz.

---

Wymienię programy na C-64 (taśma, dysk). Jerzy Raczyński, Szczecin, ul. Sopocka 1 m. 1.

---

Super komkurs dla posiadaczy C-64. Do wygrania ponad 500.000zł. Szczegóły: koperta + znaczek. Szczecin, ul. Farna 1, skr. 0632/91.

---

COMMODORE AMIGA. Gry, programy użytkowe, dema. Dużo nowości - bardzo tanio. Wysyłka pocztą, katalog gratis. Zamawiać: PAWEŁ BERSKI - Olszowa 187, 63-600 Kępno.

---

Amiga 500, Commodore 64 - wymiana programów. Koperta + znaczek. Paweł Witek, ul. Karłowicza 45/55, 58-506 Jelenia Góra.

---

PIERWSZ POLSKA PROFESJONALNA BAZA DANYCH NA C-64 - 128. Cena 20.000zł, za zaliczeniem pocztowym. T. Ciężki, ul. Koniecpolskiego 23, 98-365 Rusiec.

---

Adresy firm komputerowych + wzory listów. 15.000zl. „PINKSOFT", ul. Podgórze 17/50, 43-300 Bielsko-Biała.

---

Wymienię programy na Amigę. Proszę o spis. Łukasz Madej, ul. Kilińszczaków 14, 75-358 Koszalin.

---

C-64C, 1541 II, Final III, gwarancja, za 3.9mln. Grzegorz Małasewicz, Gdynia, tel 20-64-87.

---

Poszukuję schematu 1541 II. Rojek Józef, Świerkle, Słowackiego 17, 46-020 Czarnowąsy.

---

KLUB KORESPONDENCYJNY FANÓW DEM zaprasza do członkowstwa (koperta + znaczek). Adres: KKFD, ul. Obrońców Kępy Oksywskiej 1, 84-200 Wejherowo.

---

**UWAGA! Jako bezpłatnych nie zamieszczamy ogłoszeń typu: sprzedam oprogramowanie, jeśli z treści jednoznacznie nie wynika ich legalność. Za treść ogłoszeń i ich wiarygodność redakcja nie ponosi odpowiedzialności.**

C-64

Jak to zostało napisane w recenzji programu - patrz nr 1/91 naszego pisma - opanowanie technik programowania własnej muzyki przy użyciu tego edytora nie jest łatwe i wymaga sporego nakładu czasu, który poświęcić należy na eksperymentowanie. W tym artykule chciałbym przedstawić Czytelnikom trochę własnych doświadczeń, które mam nadzieję pomogą w lepszym wykorzystaniu tego edytora. Część zagadnień, które niepokoiły naszych Czytelników, zostały

# Doświadczenia z Voicetracker'em

Tak więc, po kolei:

a) demonstracje muzyczne, nagrane wraz z edytorem na dyskietce lub taśmie magnetofonowej są gotowe do wykorzystania w naszych własnych programach, gdyż zawierają całą procedurę odtwarzającą muzykę wraz z kompletnymi danymi utworu. Aby ją w swoim programie uruchomić, wystarczy napisać procedurę obsługi przerwań, pamiętając, że pierwszy rozkaz w języku maszynowym demonstracji muzycznej jest skokiem do procedury, która przygotowuje player do odtwarzania muzyki; następny skok należy wywoływać z naszej procedury przerwań.
Oto przykład procedury przerwań, przy pomocy której możemy skorzystać z muzyki napisanej pod Voicetracker'em. Założono, że player znajduje się pod adresem $1000 (4096 dziesiętnie).

```
        *=  $0f00      ;adres  poczatku  assemblacji

        sei
        lda  #$7f
        sta  $dc0d
        lda  #$01
        sta  $d01a
        sta  $d019
        lda  $d011
        and  #$7f
        sta  $d011
        lda  #$ff
        sta  $d012
        lda  #<irq
        ldx  #irq
        sta  $0314
        stx  $0315
        jsr  $1000     ;wywolanie  pierwszego  skoku
                       player'a
        cli
        rts

irq     jsr  $1003     ;wywolanie  drugiego  skoku
                       player'a
        lda  #$01
        sta  $d019
        jmp  $ea31
```

Demonstracje muzyczne można również wywoływać bezpośrednio po ich wgraniu do pamięci. Player zawiera bowiem krótką procedurę odtwarzającą, która w prosty sposób pozwala na odsłuchanie wgranej demonstracji. Należy postępować w ten sposób: wgrać wybraną demonstrację rozkazem LOAD "nazwa",8,1 dla dysku lub LOAD "nazwa",1,1 dla magnetofonu, po czym należy wpisać SYS 4102 i wcisnąć RETURN. Przerwać odtwarzanie muzyki możemy w dowolnym momencie przez wciśnięcie klawisza SPACE.

b) Do czego służą makroinstrukcje?
Dzięki makroinstrukcjom możliwe jest ingerowanie w kształt fali dźwiękowej podczas generowania danego dźwięku przez układ muzyczny SID, w który jest wyposażony Commodore 64. Rozwiązanie takie pozwala na generowanie dość skomplikowanych dźwięków, jak na tak prosty generator.
W Voicetrackerze problem rozwiązano w ten sposób, że każdej definicji instrumentu można przypisać makroinstrukcję, której wykonywanie rozpoczyna się każdorazowo przy każdym użyciu danego instrumentu, co pozwala nam definiować takie dźwięki jak uderzenia perkusji, symulowanie akordów (czyli tzw. arpeggio) i inne.

c) Jak relokować muzykę, gdy nie mamy do dyspozycji magnetofonu?
Z tym jest trochę więcej kłopotów, ale jest pewna droga obejścia... Niestety, jest ona dostępna tylko dla tych, którzy posiadają monitor pamięci (np. w cartridge'u).
Procedura wygląda następująco:

- wgrać do edytora muzykę, którą chcemy relokować pod inne adresy w pamięci;
- po wgraniu należy wejść do menu procedur obsługi magnetofonu i stacji dysków oraz wybrać opcję relokacji;
- należy ustalić adres relokacji i wcisnąć RETURN;
- w czasie, gdy komputer pyta nas o podanie tytułu do nagrania player'a na dyskietkę, należy wcisnąć przycisk RESET (w cartridge'u lub dorobiony własnoręcznie);
- następnie trzeba wejść do monitora pamięci i przegrać fragment od adresu $1000 do adresu, pod którym znajduje się napis „*** end of music ***".

Taka kolejność działań pozwoli nam na nagranie na taśmę player'a z przerelokowanymi adresami. Pozostaje już tylko wgrać go pod wybrany przy relokacji adres i uruchomić.

W.S.




Dziś na początek - program „taki sobie" (Marka z Grajewa).

```
10   INPUT "JAK MASZ NA IMiE?";N$
20   PRINT  "WITAJ",N$
30   INPUT  "GDZIE  MIESZKASZ";N$
40   PRINT "JA TEZ MIESZKAM W ",N$
50   END
```

Po tym powitaniu prezentujemy list Piotra z Gdańska.

**Szanowna redakcjo!**

*W listopadowym numerze „64 plus 4" w rubryce „Programy ciekawe, zwariowane i takie sobie" znalazłem procedurę „Płynne przesuwanie napisów". Nie jest to jednak najszczęśliwsze rozwiązanie tego problemu.*

*Oto moja wersja przewijająca tekst u góry ekranu:*

```
10   PRINT  CHR$(147)
20   READ  A$:IF  A$=""THEN  RESTORE
30   FOR   A=1  TO  73:B$=MID$(A$,A,1)
40   IF  B$=""THEN  GOTO  20
50   PRINT  CHR$(19);CHR$(29);CHR$(20);CHR$(17);
     CHR$(157);B$
60   FOR  I=0  TO  50:NEXT
70   NEXT  A:GOTO  20
100  DATA  "COMMODORE  64+4  &  AMIGA
     PRZEDSTAWIA:   PRZEWIJANIE   NAPISOW"
110  DATA  "AUTOR:  PIOTR  LENDZION"
999  DATA
```

*Dodatkową zaletą tego programu jest to, że na ekranie, poniżej przewijanego tekstu, mogą być umieszczone inne napisy lub rysunek, co w przypadku procedury p. Zaczko było niemożliwe.*

**Piotr Lendzion - Gdańsk.**

Kolejny list - tym razem Pawła z Lubuska.

*[...] Jest to zegar czasu rzeczywistego*

```
10   REM   ZEGAR   CZASU   RZECZYWISTEGO
20   REM   (C)  PAWEL  GOTLIB  7.12.91
30   REM   DLA  "C-64  PLUS  4  &  AMIGA"
40   PRINT  " ":REM  SHIFT  +  HOME
50   PRINT  "GODZ.";:INPUT   G
60   PRINT  "MIN.";:INPUT   M
70   PRINT  "SEK.";:INPUT   S
80   S=S+1
90   IF  S=60  THEN  M=M+1
100  PRINT  " "
110  PRINT  G;":";M;":";S
120  IF  S>59  THEN  S=0
130  IF  M>59  THEN  M=0
140  IFM=59   AND   S=59   THENG=G+1
150  FOR  I=1  TO  600
160  NEXT  I
170  GOTO  80
```

*Program można rozwinąć przez wprowadzenie alarmu. Pewną wadą programu jest to, że godziny „lecą" w nieskończoność, ale i to można naprawić.*

Mariusz z Łomży napisał do nas tak:

*[...] zainteresowała mnie rubryka „Programy ciekawe, zwariowane i takie sobie". Postanowiłem napisać program, który zrozumiałby przeciętny początkujący. Moje dzieło nie jest niczym nadzwyczajnym, a zresztą, przekonajcie się sami...*

```
0    REM  STOPER  (C)  BY  MARIUSZ  NOWIK
10   PRINT   CHR$(147)
20   IF  B=60  THEN  B=0:A=A+!
30   B=B+1
40   IF  A=60  THEN  A=0:G=G+1
50   IF  G=24  THEN  G=0:A=0
60   PRINT   "STOPER":PRINT
70   PRINT   G;":";A;":";B
80   FOR  I=1  TO  290:NEXT  I
90   PRINT   CHR$(147)
100  PRINT   "STOPER":PRINT
110  PRINT  G;"  ";A;"  ";B
120  FOR  I=1  TO  290+NEXT  I
130  GOTO  10
```

Grzegorz z Ropczyc tak pisze w swoim liście:

*[...] mój program nie jest zwariowany, ani nie sądzę żeby był ciekawy. Jest po prostu taki sobie. Może być on pomocny dla tych, którzy chcą wygrać w dużym lotku. Działa on następująco: wybiera sześć liczb z zakresu od 1 do 49, następnie wyświetla je na ekranie i podaje komunikat: „JEŚLI CHCESZ UZYSKAĆ KOLEJNE SZEŚĆ LICZB, NACIŚNIJ KLAWISZ '↑'.*

*Po naciśnięciu tego klawisza komputer wyświetli na ekranie następne sześć liczb i znowu ten sam komunikat, i tak w kółko.*

*Oto listing programu:*

```
10   PRINT  CHR$(147):FOR  S=1   TO6:A=INT
     (RND(1)*49+1):PRINT:PRINT     A
20   NEXT:PRINT:PRINT  "JAK  CHCESZ  UZYSKAC
     KOLEJNE  SZESC  LICZB,  NACISNIJ  KLAWISZ  '↑'"
30   GET  Q$:IF  Q$<>"↑"  THEN  30
40   GOTO10
```

*Niestety, posiada on jedną wadę, a mianowicie czasami wyświetli dwie takie same liczby w jednej szóstce. Niestety nie wiem, jak sobie z tym poradzić. [...]*

Wszystkim Autorom serdecznie dziękujemy i zapraszamy Czytelników do dalszego, wspólnego redagowania tej rubryki.

**W.S.**

C-64

Po małej przerwie zapraszamy do kontynuowania nauki assemblera 6510. Dziś poznamy wszystkie te rozkazy, które dotąd nie były omawiane.

# ASSEMBLER 6510 - lekcja 6

**ROL (rotate one bit left) - obrót o jeden bit w lewo.**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | * |

| Rozkaz | Kod | Cykle |
|---|---|---|
| ROL | 2A | 2 |
| ROL $nn | 26 | 5 |
| ROL $nn,X | 36 | 6 |
| ROL $nnnn | 2E | 6 |
| ROL $nnnn,X | 3E | 7 |

Rozkaz ten powoduje obrót bitów w danym bajcie o jeden bit w lewo. Przy czym zawartość znacznika C przesuwa się do zerowego bitu, a siódmy bit przesuwa się do znacznika C.

Oto prosty przykład ilustrujący działanie tej instrukcji:

01011001 = $59 i C=1

Po obrocie w lewo:

1011001 = $B3 i C=0

Aby to sprawdzić uruchomimy krótki programik:

```
1000 LDA #$58
1002 ROL
1003 BRK
```

W akumlatorze otrzymujemy $B3, a w C znajduje się 0. Widać więc, że wszystko się zgadza.

**ROR (rotate one bit right) - obrót o jeden bit w prawo.**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | * |

| Rozkaz | Kod | Cykle |
|---|---|---|
| ROL | 6A | 2 |
| ROL $nn | 66 | 5 |
| ROL $nn,X | 76 | 6 |
| ROL $nnnn | 6E | 6 |
| ROL $nnnn,X | 7E | 7 |

Rozkaz ten działa podobnie do ROL, ale obrót bitów odbywa się w prawą stronę.

**DEC (decrement memory) - zmniejszenie zawartości komórki pamięci o jeden.**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | - |

| Rozkaz | Kod | Cykle |
|---|---|---|
| DEC $nn | C6 | 5 |
| DEC $nn,X | D6 | 6 |
| DEC $nnnn | CE | 6 |
| DEC $nnnn,X | DE | 7 |

Rozkaz ten powoduje zmniejszenie zawartości danej komórki pamięci o jeden.

Działanie jego jest bardzo proste, więc przedstawiam tylko krótki przykład:

```
1000 LDA #$DE
1002 STA $1000
1005 DEC $1000
1008 LDA $1000
100B BRK
```

Po wykonaniu tego programu otrzymamy wartość $DD w akumulatorze. Warto zwrócić uwagę na to, iż nie istnieje rozkaz działający na samym akumulatorze.

**DEX (decrement X register) - zmniejszenie zawartości rejestru X o jeden.**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | - |

| Rozkaz | Kod | Cykle |
|---|---|---|
| DEX | CA | 2 |

Rozkaz ten działa jak DEC, ale zmniejsza zawartość rejestru X.

**DEY (decrement Y register) - zmniejszenie zawartości rejestru Y o jeden.**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | - |

| Rozkaz | Kod | Cykle |
|---|---|---|
| DEY | 88 | 2 |

Rozkaz ten działa jak dwa pozostałe, ale operuje na rejestrze Y.

**INC (increment memory) - zwiekszenie zawartości komórki pamięci o jeden.**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | - |

| Rozkaz | Kod | Cykle |
|---|---|---|
| INC $nn | E6 | 5 |
| INC $nn,X | F6 | 6 |
| INC $nnnn | EE | 6 |
| INC $nnnn,X | FE | 7 |

Rozkaz ten zwiększa zawartość określonej komórki o jeden.
Ilustruje to krótki przykład:

```
1000 LDA  #$10
1002 STA  $FB
1004 INC  $FB
1006 LDA  $FB
1008 BRK
```

Wynikiem działania tego programu będzie liczba $11 w akumulatorze. Podobnie, jak rozkaz DEC, tak i INC nie ma możliwości operowania na akumulatorze.

**INX (increment X register) - zwiększenie zawartości rejestru X o jeden.**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | - |

| Rozkaz | Kod | Cykle |
|---|---|---|
| INX | E8 | 2 |

Rozkaz działa jak INC, ale operuje tylko na rejestrze X.

**INY (increment Y register) - zwiększenie rejestru Y o jeden**

| N | V | D | I | Z | C |
|---|---|---|---|---|---|
| * | - | - | - | * | - |

| Rozkaz | Kod | Cykle |
|---|---|---|
| INY | C8 | 2 |

Na koniec proponujemy kilka zadań:

1) Napisać procedurę:
   a) mnożącą
   b) dzielącą dowolną liczbę dwubajtową przez osiem.
2) Napisać procedurę sumującą liczby od 0 do 255. Wynik ma być zapisany w dwóch komorkach pamięci na stronie zerowej.
3) Napisać procedurę przepisującą zadany obszar pamięci (podany jest adres początku i końca obszaru) w podane miejsce. Przy czym możliwość nałożenia się obszarów źródłowego i docelowego można pominąć.

To już wszystko na dziś. Za miesiąc poznamy rozkazy porównań, przesłań pomiędzy rejestrami oraz odkładania rejestrów na stos. Wkrótce też, po skończeniu omównienia listy rozkazów zajmiemy się zastosowaniem wszystkich wiadomości w praktyce. Czekam również na ewentualne propozycje Czytelników dotyczące tego działu.

JARRI




**UWAGA!**

**Otrzymaliśmy sporo listów poruszających problem klubu komputerowego Amiga z Bełchatowa. Przytaczamy dwa z nich, mając nadzieję, iż organizatorzy klubu ustosunkują się do nich. Wymowa tych listów jest jednoznaczna i nie wymaga komentarza.**

**Droga Redakcjo!**

*W majowym numerze Waszego miesięcznika została zamieszczona reklama klubu komputerowego z Bełchatowa. Zwiedziony zaskakująco korzystną ofertą, napisałem. Otrzymałem bardzo szybko odpowiedź.*

*Wysłałem cztery dyski z programami, za które miałem otrzymać trzy dyski z programami i katalog oprogramowania klubu na dysku. Oczywiście uczciwa zamiana, lecz na mojej przesyłce uczciwość się kończy. Czekam już od wakacji i nie otrzymałem swojej własności.*

*Pisałem do klubu. Bez skutku.*

*Przesyłam Wam pokwitowanie przekazu pieniężnego na kwotę, która była warunkiem przyjęcia do klubu, oraz potwierdzenie wysłania przesyłki.*

*W liście tym znajdziecie także odpowiedzi klubu na moje listy.*

*Nie piszę tego listu, aby wyrzucać Wam to, że zamieszczacie takie ogłoszenia. Oczywiście nie możecie wiedzieć, czy ogłaszające się firmy, wywiązują się z umów. Piszę ten list, dlatego aby przestrzec innych użytkowników Amigi.*

*Na koniec smutna konkluzja. Miałem wcześniej Atari 130XE, Amigę mam od kwietnia. Wiele słyszałem o bezinteresowności Commodorowców (być może nie odnosi się do Amigowców). Jedno jest pewne! Wśród Atarowców, których znałem byli prawie sami tacy, którzy w komputerze widzieli tylko źródło zarobku, ale nie spotkałem się nigdy z kanciarzami.*

*Incydent z klubem z Bełchatowa nauczył mnie, aby nie załatwiać takich spraw listownie (gdy miałem Atari, to był jedyny sposób na pozyskanie literatury).*

*Próbowałem zakupić literaturę na Amigę drogą pocztową, cóż z tego jeśli firmy, do których pisałem nie odpowiadają.*

*Z poważaniem*

**Wojtek „Viper" [illegible]**

**Szanowna Redakcjo!**

*[...]*

*Zainteresowało mnie to ogłoszenie, ponieważ posiadam Amigę 500. Wysłałem do Bełchatowa list z moją chęcią wstąpienia do klubu. Po niedługim czasie otrzymałem list z kartką (którą załączam do listu), oczywiście wpłaciłem podaną kwotę pieniędzy i czekałem na odpowiedź. Przyszła po kilku dniach dyskietka za którą, oczywiście też musiałem zapłacić. Na dyskietce był aktualny spis. Wybrałem interesujące mnie programy i wysłałem 8 dysków do Bełchatowa...*

*I tu się kończy historia o dyskach.*

*Już więcej nie usłyszałem, ani nie otrzymałem wiadomości na moje kolejne 2 listy. [...]*

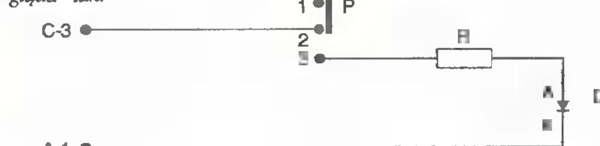
**I. Biliński**

**DATASETTE**

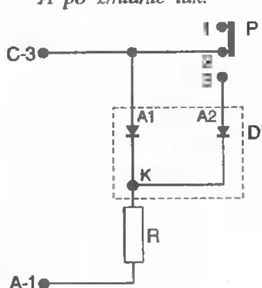
Pan Adam Komorowski z Wrocławia nadesłał propozycję usprawnienia magnetofonu do C-64.

*[...] Brakowało mi tam sygnalizacji pracy silnika w czasie ładowania programów. Jak wiadomo, czerwona dioda w tym urządzeniu świeci tylko w czasie zapisu. W prosty sposób można tam zainstalować diodę dwukolorową, która świeci na zielono w czasie ładowania programu, a na czerwono w czasie zapisu.*

*Schemat zasilania tej diody przed wprowadzeniem zmian wygląda tak:*




*A po zmianie tak:*




*Dioda zielona ma większy spadek napięcia niż czerwona, dlatego gdy przeł. P zwiera styki 2 i 3, to świeci tylko czerwona.*

*D" to dioda dwukolorowa o wspólnej katodzie, bezpośrednio do C-3 podłącza się anodę diody zielonej, do przełącznika zapisu anodę diody czerwonej.[...]*

Przeróbka nie jest skomplikowana i nawet początkujący amatorzy - elektronicy mogą ją wykonać.

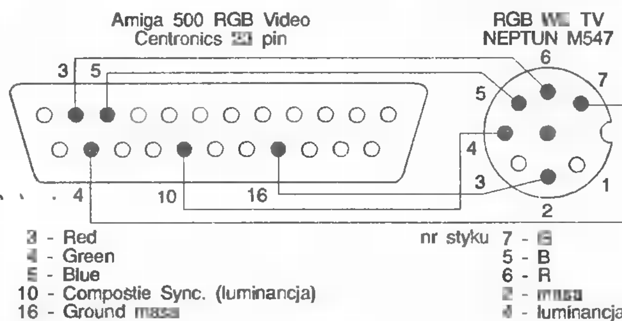
Jak zawsze zwracamy uwagę na ostrożność i dokładność w czasie pracy!

**Amiga i Neptun M547**

Publikujemy fragment listu pana Bartusia Marka z Nowego Bierunia.

*[...] Podłączyłem sobie Amigę do telewizora Neptun M547 (wciskano mi na siłę modulator) efekty podłączenia okazały się wspaniałe.*

*Podłączenie wykonałem 4-żyłowym kabelkiem z ekranem, kłopoty miałem tylko z gniazdem centroniks 23pin (przerobiłem z gniazda 25pin). [...]*




Mamy nadzieję, że zainteresuje on posiadaczy Amigi i odbiornika TV Neptun M547.

Do naszej redakcji dotarł nietypowy list. Pani Alicja z Poznania samotnie wychowuje chorego syna. Nie będziemy przytaczać szczegółów listu - jest on bardzo osobisty.

Jej syn otrzymał w prezencie Commodore plus 4. Niestety nie posiada, ani instrukcji, ani żadnych innych materiałów, które pozwoliły by Mu opanować ten komputer. Wielu Commodorowców zaczynało swoją „karierę" od C-16, plus 4. Być może posiadają niepotrzebne już materiały (najlepiej w j. polskim), które mogliby bezinteresownie odstąpić choremu chłopcu. Wierzymy, że nasi Czytelnicy nie zawiodą. Materiały prosimy przesłać na adres redakcji, wszystkie przekażemy do Poznania.

Dziękujemy.

# PUBLIC DOMAIN PACK

## PUBLIC DOMAIN PACK C - 64

### Styczeń

*STRONA A*

- Mega demo grupy „VISION"-MIST2

*STRONA B*

- Preview do gry „UN SQUADRON"
- Preview do gry „PUZZLENOID"
- Preview do gry „TURRICAN"

### Luty

*STRONA A*

- TUNE OF MONTH
- LOGO WRITER V 2.0
- FAST CRUELCRUNCH
- WRATH+(DEMO)[02]
- DREPTACZ _ BASIC

*STRONA B*

- SWISS CHEESE/CFA
- DISK FAST LOADER

### Marzec

*STRONA A*

- FONT GRUB 1.0
- PROJEKTANT DUSZKÓW
- STRZAŁKA 64+
- PIRATEK - GRA
- V4.0 - SYMPHONIES
- CRUISER
- THE FIRST
- COMMERCIAL BREAK
- RELAKATOR 64
- KOREKTOR 64
- FLASH

*STRONA B*

- HOT SHOT nr9 (zach. magazyn fanów)
- BAD NEWS nr2 - j.w.
- DEMO - rekord - 290 SPRITE'ów!
- DEMO: NEW INTRO
- DEMO: LET'S DYCP
- KONTAKT CORNER _ adresy, kontakty
- NEW FAST - działa z 1541 I 1541 II
- CSLINKER V2.0

### Kwiecień

*STRONA A*

- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku

*STRONA B*

- „ONE YEAR - RADIUS" - mega demo grupy RADIUS. Bardzo ładna grafika

### Maj

*STRONA A*

- CRUEL SOLIDERS - demo
- DESTINATION - demo
- SUCKER DJ! - demo (digi mix)
- MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów

*STRONA B*

- MEGA DEMO „INFOSYSTEM 91"

### Czerwiec

*STRONA A*

- FONTEDITOR
- SINDATA EDITOR
- COLOR EDITOR
- DISK - NOTER
- GWIAZDY - demo graficzne
- FILGRAEPH 2.2/BML
- NOTE TO FLI V 2.2
- AFLI - EDITOR V 1.2
- RESET - MON,8,1
- TURBO - ASS 5
- ...HIGHLIFE #5
- AXEL NEWS #1
- DISK NOTKA/PADUA

*STRONA B*

- PSC - MAG #9'06/91
- CONSPIRE?OREGON - demo
- CONTACT DEMO/ORE
- SHOWPIX

### Lipiec

*STRONA A*

- Mega demo „MY, OH MY!" grupy LIGHT

*STRONA B*

- Game Music Composer - edytor muzyczny grupy GRAFFITY Węgier.

### Sierpień

*STRONA A*

- MegaDemo „Unnamed" grupy CAMELOT
- Sound Killer - edytor muzyczny grupy TOPAZ
- AFLI - Editor graficzny techniki A-FLI
- Disk-Dos obsługa komend stacji dysków
- Noter v2.2 grupy TOPAZ
- IFFL - Squeezer kompresor dyskowy
- Dismaster+ - edytor do dyskietek
- Super Copy - DOS szybki program kopiujący do zbiorów

*STRONA B*

- Mega Demo fińskiej grupy TOPAZ - „Graveyard Blues"

### Wrzesień

*STRONA A:*

- Mega Demo grupy FLASH

*STRONA B:*

- Hot Shot - magazyn dyskowy
- Code Sucker monitor - pr. użytkowy grupy PADUA
- Mountain Ride - gra w BASIC

### Październik

*STRONA A i B:*

- MEGA DEMO „AIRDANCE 4" grupy T.A.T.

### Listopad

*STRONA A*

- NEW LAW & ORDER!
- FLT/LEGOLAND
- FLT/LEGONOTE
- TERMINAT.2%/FLT

*STRONA B*

- SM. CRIMINAL #08
- SMALL BUT FINE
- HOLLY SMOKE/M12
- UNITE!/SYLVIO

### Grudzień

*STRONA A*

- ARMAGEDDON 3
- NOTE TO DEMO

*STRONA B*

- OUTRUN 2 MUS $ SFX
- AFTERBURNER/MON
- TRIVIA-GAME MUSIC
- FORM.I. SIMULATOR
- 2400AD END-TUNE
- NIGHTHUNTER DIGI
- ELIMINATOR MUSIC
- TOMCAT MUS./MON
- ZAMZARA TUNE/MON
- NOTE TO DISK
- HIGHLIFE#9

## PUBLIC DOMAIN PACK AMIGA

### Styczeń

- Programy kompresorów danych
- Grafiki Borysa Vallejo
- Prezentacja najlepszych muzyczek
- INTUITRACKER

### Luty

- Request player; Multi ripper
- 3-rd day; Phantasmagoria - demo
- Master Seka; Virus Ekspert v1.6
- AMOS-programy; Moduły: Killing game show, Upon Me, Let's swing it.

### Marzec

- Najnowszy i najlepszy program muzyczny PROTRACKER V1.0 (pakiet programowy)
- Najlepsze muzyczki: NOW WAIT? - DR.AWESOME
- AMOS - procedury DEMO grupy REBELES „TOTAL TRIPLE TROUBLE"

### Kwiecień

- RUBBER VECTORS - demo
- KEFTALES - demo
- DISK MASTER V3.0
- Moduły muzyczne:
  > TECHNOSTYLE 2
  > GALAXY 2
- GRAFIKA - prezentujemy rysunki
  > RICK PARKS

### Maj

- VIRUS X 5.0
- VIRUS TERMINATOR
- PARADOX - demo
- STORMCHILD - demo
- Moduły muzyczne:
  > MIAMI VOICE
  > ANTI ATARI SONG

### Czerwiec

- POWER BOOT - własne menu dysku
- DISK CODING SYSTEM - program do zabezpieczania dysków
- Konwerter IFF - ANSI
- AUER NATION - demo
- Moduły muzyczne
- DOCS - opis gry ELWIRA
- LAMER DEFENCE - do wykrywania i niszczenia wirusów
- REWENG GO OF THE LAMER - grafika w trybie D_HAM

### Lipiec

- Sanity - demo
- Amiga - Tanx (1Mb) - gra
- Little Beau (1MB) - gra
- There is A Light/Tonid - modules

### Sierpień

- Real 3D - demo nowego programu do raytraceing'u
- Moduł Muzyczny XTC STEREO

### Wrzesień

- MODUŁY MUZYCZNE dla programu TFMX:
  > R - TYPE
  > THE HOUSE OF TECHNO
- VIRUS EXPERT v181 + 143
- BOOT BLOCK'I
  > BOOTX v 3.80
  > IMPLODER v 4.0

### Październik

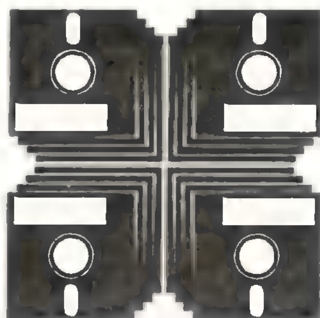
- ANARCHY - „THE INSPIRATION IS NONE"
- DUAL CREW - „NEW DIMENSION"
- SANITY - „ELYSIUM"

**Listopad**

- COMPUTER HEAD - animacja
- CONFUSED - MODUŁY POD MEDPLAYER I WIELE SAMPLI
- ROCKED -
- SAVE GAME „MONKEY ISLAND"

**Grudzień**

- GEM X
- BOOTX V. 4.13
- FINAL KIT -monitor
- MEGA-MON
- VARIA



**PUBLIC DOMAIN PACK C-64
TAPE NR 1**

- TURBO
- SINUSDATA - EDITOR
- FAST CRUNCHER V3
- ANAL S.C. IBEYOND
- VECTOR - VICTORY
- PUZZLENOID+4
- TUNE OF MONTH #1
- NIM
- STRZAŁKA 64+
- LOGO - WRITER V.2.0
- CAN'T TOUCH IKU!
- NTRO PRV
- BONZIEED!!
- ZAX PACKIS
- READ THIS FIRST
- COMMERCIAL BREAK
- 250 SPRITES!
- NOTE - ABOUT
- BAD NEWS NR2
- TO BAD NEWS...
- CONTACT CORNER!
- PROJEKT DUSZKÓW
- SYMPHONY NR14
- SYMPHONY NR15
- SYMPHONY NR16
- SYMPHONY NR17
- SYMPHONY NR18
- SYMPHONY NR19
- CRUISER/GIANTS
- NOTE>ANO<PADUA
- LET'S DYSP!
- FINALTAPE
- MUSIC - SEARCHER

**PUBLIC DOMAIN PACK C-64
TAPE NR 2**

- TURBO
- PUBL. DOMAIN. INFO
- FONTGRUB 1.0
- DREPTACZ BASIC
- LOAD DIS FIRSY
- MACROASSEMBLER
- TURBOASSEMBLER
- RELOCATOR
- LOGOPAINTER II
- REASSEBLER
- SPRITE - EDITOR
- FAST - CRUEL U.2.5
- HIGHLIFE NR5
- AXEL NEWS NR1
- GWIAZDY
- FLIGRAPH 2.2/BML
- NOTE TO FLI V.2.2
- DISKNOTKA/PADUA
- MEGA PACKER/T
- MIST II/ VISION
- TTECHSCR &DYSP
- PLASMA - WORLD
- VECTORBOBS...
- VECTOR - PLOTS
- FLI - UPSCROLL
- BORDER - HIRES
- ROCK AROUND
- FACEWRITER
- CHAR EDIT 2+2
- DISKNOTER
- DESTINATION'91
- CONTACTDEMO/ORE
- FONTEDITOR
- THE END

**PUBLIC DOMAIN PACK C-64
TAPE NR 3**

- TURBO
- PUBLIC DOMAIN NOTE
- GRAVEYARD NOTES!
- NOTE FROM BEAT!!
- ANONYM SPEAKING!
- SNDK. V3.7/TOPAZ
- AFLI - EDITOR
- NOTER V2.2/TOPAZ
- DLW V1.5/TOPAZ
- CODE - S.MON/PADUA
- OPINION - POLL/PDA
- MOUNTAIN RAID
- PART 1
- PART 2
- PART 3
- PART 4
- PART 5
- FAIRLIGHT 1
- FAIRLIGHT 2
- FAIRLIGHT 3
- FAIRLIGHT 4
- FAIRLIGHT 5
- THE END

**PUBLIC DOMAIN PACK C-64
TAPE NR 4**

- TURBO
- OUT RUN 2 MUS & SFX
- AFTER BURNER/MON
- FORM.1.SIMULATOR
- 2400 AD.END - TUNE
- NIGHT HUNTER DIGI
- ELEMINATOR MUSIC
- TOMCAT MUSIX/MON
- ZAMZARA TUNE
- DYNAMIX TUNE
- HIGHLIFE NR9
- SNAKES C3
- SNARK C3
- SNERD C3
- WAREHOUSE C3
- STARTREK C3
- TOWER
- SNOOPY
- NEW LAW & ORDER
- FLT/LEGONOTE...
- TERMINAT.2%/FLT
- UNITE!/SYLVIO
- BALL - SCOPE/451
- TRIVIA - GAME MUS.
- RESET - MONITOR
- HOLY SMOKE

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr: 5.09011-400522.7-2511-30-111.0 następujące kwoty: 20.000zł za pojedyńczy zestaw dyskowy dla C-64, 30.000 zł za zestaw programów PD na kasecie, 25.000zł za zestaw dla Amigi.

Blankiety wpłat powinny być CZYTELNIE wypełnione i zawierać: **imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64D"** - jeśli zamawiamy zestaw dla C-64 na dyskietce lub **„PDP-64T"** - dla zestawu taśmowego, zestaw dla Amigi prosimy zaznaczać skrótem **„PDP-A"** - dane te prosimy umieszczać **na wszystkich** odcinkach dowodu wpłaty.

W prenumeracie zestawy kosztują: **PDP-64 - 18.000zł** (12 numerów 216 tys. zł), **PDP-A - 22.000 zł** (12 numerów 264 tys. zl). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Powyższe warunki odnoszą się również do naszych zestawów wydanych w 1991r .

Zestawy taśmowe PDP-64 w 1992r. będą ukazywały się w miarę napływu nowych, ciekawych programów - o czym będziemy informować na łamach naszego pisma.

# Zamów nie zwlekaj!

# VOICETRACKER V4.0

# C-64

# Rewelacyjny program muzyczny!

Tylko 50.000 zł kosztuje fantastyczny edytor muzyczny wykorzystujący ogromne możliwości dźwiękowe komputera Commodore - 64. Oferowany zestaw zawiera dyskietkę lub taśmę magnetofonową z programem VOICETRACKER V4.0, instrukcję obsługi, oraz - dodatkowo - przykładowe demonstracje muzyczne. **UWAGA! Wersja magnetofonowa tylko 40.000 zł**

Przedsiębiorstwo ABUK posiada wyłączność na dystrybucję tego programu. Wszelkie kopiowanie programu i powielanie instrukcji jest zabronione. Nabywcy otrzymują rejestrowane kopie programu wraz z prawem nabywania nowych wersji po znacznie obniżonych cenach oraz wymiany dyskietki w razie uszkodzenia. Studiom komputerowym proponujemy zakup hurtowy (przy zakupie powyżej 10 kompletów udzielamy 20% rabatu).
Chcąc stać się posiadaczem programu VOICETRACKER V4.0 wystarczy dokonać wpłaty 50.000 zł (wersja dyskowa) lub 40.000 zł (taśma) na konto: Bank PKO SA Bydgoszcz, konto nr 5.09011-400522.7-136-11-111.0. Na blankiecie prosimy czytelnie podać swoje imię, nazwisko i adres wraz z dopiskiem „VV4.0" uzupełnionym literką „T" - taśma lub „D" - dyskietka.

W związku z pojawiającymi się kłopotami w dystrybucji oferowanych przez nas dyskietek i taśm (wynikającymi z nieczytelnego bądź niekompletnego wypełnienia blankietów wpłat) przedstawiamy obok specjalny druk. Blankiet ten może służyć jako zamówienie i dowód wpłaty dla wszystkich oferowanych przez nas usług: sprzedaż dyskietek i taśm PDP, Voicetracker'a, zamówienie ogłoszeń itd.

REDAKCJA

---

Odcinek dla wpłacającego

Zł .................
słownie .................
.................
wpłacający .................
.................
.................
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.**
**87-200 Wąbrzeźno, ul. 1 Maja 33,**
**Bank PKO SA Bydgoszcz, konto:**
**5.09011-400522.7-136-11-111.0.**

Opłata
zł.........

---

Odcinek dla Banku

Zł .................
słownie .................
.................
wpłacający .................
.................
.................
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.**
**87-200 Wąbrzeźno, ul. 1 Maja 33,**
**Bank PKO SA Bydgoszcz, konto:**
**5.09011-400522.7-136-11-111.0.**

Opłata
zł.........

---

Odcinek dla posiadacza rachunku

Zł .................
słownie .................
.................
wpłacający .................
.................
.................
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.**
**87-200 Wąbrzeźno, ul. 1 Maja 33,**
**Bank PKO SA Bydgoszcz, konto:**
**5.09011-400522.7-136-11-111.0.**

Opłata
zł.........

---

Odcinek dla Poczty

Zł .................
słownie .................
.................
wpłacający .................
.................
.................
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.**
**87-200 Wąbrzeźno, ul. 1 Maja 33,**
**Bank PKO SA Bydgoszcz, konto:**
**5.09011-400522.7-136-11-111.0.**

Opłata
zł.........

**W związku ze znacznym wzrostem opłat za przesyłki pocztowe, mając na uwadze dobro naszych czytelników sugerujemy zamawianie numerów zaległych naszego czasopisma na nieco innych zasadach.**

Koszty przesyłki za tzw. zaliczeniu pocztowym przedstawia tabela (rubryki 2, 3, i 4). Wynika z niej, że koszty przesłania dwóch pierwszych numerów „64 plus 4" są większe niż ich wartość! Zamawiając numery wartości 6.000 zł zmuszeni jesteście dopłacić poczcie jeszcze 8.000 zł!

Zupełnie inaczej przedstawia się sytuacja w tabelach 5, 6 i 7. Wpłata kwoty z rubryki nr 7 na nasze konto (wraz z czytelną adnotacją, których numerów dotyczy, umieszczoną na wszystkich odcinkach blankietu) powoduje, że otrzymujecie przesyłkę bez kosztów pobrania!

Proponujemy abyście zaległe numery naszego pisma zamawiali w sposób następujący: wykorzystując tabelę - rubryki 1, 2, 5 i 7 - wyliczyli kwotę wpłaty, a następnie przesłali ją na nasze konto. Po otrzymaniu wpłaty natychmiast realizujemy przesyłkę!

**Uwaga:** dla dokonania wpłaty prosimy wykorzystać blankiet zamieszczony na tej stronie. Wszystkie dane prosimy pisać czytelnie i - zgodnie z rubrykami na blankiecie - **dziękujemy!**

| Numery | Za pobraniem | | | Wpłata na konto | | |
|---|---|---|---|---|---|---|
| | Wart. pisma | Koszt wysyłki | Razem | Wart. pisma | Porto | Razem |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| XI,XII | 6.000,- | 8.000,- | 14.000,- | 6.000,- | 1.000,- | 7.000,- |
| XI,XII,I | 11.000,- | 8.500,- | 19.500,- | 11.000, 1.500,- | 12.500,- | |
| XI,XII,I,II | 16.000,- | 8.500,- | 24.500,- | 16.000,- | 1.500,- | 17.500,- |
| XI,XII,I,II,III | 21.000,- | 8.500,- | 29.500,- | 21.000,- | 1.500,- | 22.500,- |
| XI,XII,I-IV | 26.000,- | 10.000,- | 36.000,- | 26.000,- | 2.000,- | 28.000,- |
| XI,XII,I-V | 31.000,- | 10.000,- | 41.000,- | 31.000,- | 2.000,- | 33.000,- |
| XI,XII,I-VI | 36.000,- | 10.000,- | 46.000,- | 36.000,- | 2.000,- | 38.000,- |
| XI,XII +I-VIII | 46.000,- | 14.000,- | 60.000,- | 46.000,- | 3.000,- | 49.000,- |



TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Virus Expert powstał w roku 1987, a jego autorem był niejaki pan Paul van der Valk. Zaiste był to program, jak na tamte czasy wspaniały, ale czasy się zmieniają, powstają nowe wirusy i potrzebne są coraz nowsze, doskonalsze programy do ich „mordowania".

AMIGA

# VIRUS EXPERT II

Dlatego w cztery lata później pojawił się Virus Expert II, którego autorem jest Maciek Marzec. Cień padł na populację wirusów...

Program Virus Expert II jest programem napisanym całkowicie od nowa w języku maszynowym, a to co łączy go z programem Virus Expert, to sposób analizy bootbloku oraz biblioteka bootbloków.

Maciek zajmował się kiedyś udoskonalaniem Virus Experta, a jego główne poprawki dotyczyły zwłaszcza analizy bootbloku, która okazała się naprawdę bardzo dobrą.

Program Virus Expert II jest spolszczony - co jest zrozumiałe ze względu na to, iż jest adresowany do polskich użytkowników i jest sprzedawany wyłącznie w Polsce (jak na razie).

Jeżeli chodzi o jego możliwości zabijania wirusów (zwłaszcza tych plikowych) to są one ogromne. Wykrywa on trzydzieści (!) wirusów plikowych co w porównaniu z innymi wysuwa go na prowadzenie (na przykład najnowszy BootX (v4.13) wykrywa tylko 29 tych wirusów). Wirusy plikowe możemy wykrywać sprawdzając programy wykorzystywane w startup-sequence, wszystkie programy na dysku bądź pojedyncze zbiory. Jedyną wadą, jeżeli chodzi o wykrywanie i usuwanie tych wirusów jest to, że po wykonaniu tej operacji dla wirusów linkujących się (dołączjących się do plików), wirus zostanie usunięty wraz z zainfekowanym plikiem.

Uważam, że usuwanie samych wirusów linkujących się jest bardzo potrzebne i że Maciek wkrótce wprowadzi tę opcję do swojego programu. Już wielokrotnie otrzymałem wiele wartościowych programów z dolinkowanymi wirusami i przyznam, że usuwanie ich za pomocą monitora nie należy do przyjemności. Natomiast jeżeli chodzi o wirusy znajdujące się na bootbloku to ich analiza w nowym Virus Expercie jest bardzo dobra potrafi on ocenić dosyć trafnie bootblok i wskazać dokładnie jakie rzeczy ten bootblok robi. Oprócz tego dołączona jest procedura bootbloków, które kodują same siebie.

Dołączona biblioteka wirusów zawiera tylko ich nagłówki, aby jakiś złośliwy posiadacz tego programu nie rozpoczął infekcji dysków wszystkim swoim mniej lub bardziej lubianym znajomym. Według mnie jest to bardzo dobre posunięcie ze strony Maćka, zwłaszcza, że teraz program już się nie myli i nie podaje błędnych nazw (np. w jednej z wersji Virus Experta (tego starego) był umieszczony w bibliotece nagłówek pewnego wirusa, który pokrywał się idealnie z nagłówkiem innego programu użytkowego, który był brany właśnie za tego wirusa).

Ogólnie oceniając program jest bardzo dobry i wart wydania stu tysięcy złotych. Pozwala na dokładną analizę i uśmiercanie wielu bardzo różnych wirusów i przewyższa w tym względzie inne programy, służące do zabijania wirusów.

Maciek powinien jeszcze - moim zdaniem - wprowadzić przy teście plików, określanie czy program jest spakowany (np. PowerPackerem lub Imploderem) i podawać te dane o programie lub ewentualnie depakować je i dopiero wtedy dokonywać testu na wirusy plikowe. Opcja ta byłaby bardzo przydatna, gdyż czasami trafiają się wirusy dolinkowane spakowane razem z programem i są wtedy niewykrywalne.

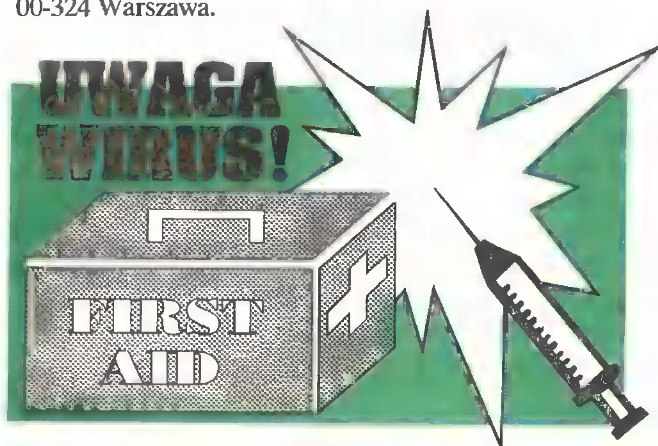
Testując ten program znalazłem jeszcze jedną rzecz, która wymaga poprawki, a mianowicie: nie rozpoznaje on czy program, który „wisi" pod wektorami jest programem użytkowym (na przykład odtwarzalne ramdyski RAD: i RRD: lub dyskowa wersja Kickstartu 2.0).

Program Virus Expert II określam jako dobry i polecam go wszystkim, którzy mają kłopoty z wirusami.

Wersja testowana: 1.201
Sprzęt: Amiga 500 (Kickstart 1.2/1.3/2.04) z 2.3 MB pamięci

**Marcin „Duddie" Dudar**

Dystrybucja: InterComp sp. z o.o., ul.Karowa 18a/20, 00-324 Warszawa.



AMIGA

Po miesięcznej przerwie, dziś kolejna porcja instrukcji procesora MC68000.

# KĄCIK POCZĄTKUJĄCEGO KODERA cz.9

**Rozkazy przesunięcia.**

- **ASL - Aritmetic Shift Left - przesunięcie arytmetyczne w lewo.**

Przesunięcie arytmetyczne w lewo jest to przesunięcie operandu o zadaną ilość bitów przy czym najstarszy bit idzie do znaczników X i C, a do najmłodszego bitu wpisywane jest zero.

**ASL #liczba,Dx**

Przesunięcie rejestru danych o liczbę z zakresu od 1 do 8. Przesunięcie może się odbywać w zakresie bajtu, słowa oraz długiego słowa.

**ASL Dx,Dy**

Przesunięcie rejestru danych Dy o liczbę zawartą w rejestrze danych Dx. Przesunięcie może się odbywać w zakresie bajtu, słowa i długiego słowa.

**ASL adres efektywny**

Przesunięcie zadanego adresu efektywnego o jeden bit w lewo. Operacja wykonuje się na słowie. Dopuszczalnymi adresami efektywnymi są: (An), (An)+, -(An), x(An), x(An,R.s) oraz komórka pamięci.

Przykłady:

```
ASL.L   #4,D3
ASL.W   D5,D2
ASL     (A0)
```

**Znaczniki:**

| | |
|---|---|
| X i C | - ustawiane zgodnie z ostatnim wysuniętym bitem |
| Z | - ustawiany gdy wynik jest równy zero |
| N | - ustawiany zgodnie z najstarszym bitem wyniku |
| V | - ustawiany gdy najbardziej znaczący bit został zmieniony podczas przesunięcia |

- **ASR - Aritmetic Shift Right - przesunięcie arytmetyczne w prawo.**

Przesunięcie arytmetyczne w prawo jest to przesunięcie operandu o zadaną ilość bitów przy czym najmłodszy bit idzie do znaczników X i C, a najstarszy jest kopiowany do samego siebie, czyli wartość po operacji nie zmienia znaku.

**ASR #liczba,Dx**

Przesunięcie rejestru danych o liczbę z zakresu od 1 do 8. Przesunięcie może się odbywać w zakresie bajtu, słowa oraz długiego słowa.

**ASR Dx,Dy**

Przesunięcie rejestru danych Dy o liczbę zawartą w rejestrze danych Dx. Przesunięcie może się odbywać w zakresie bajtu, słowa i długiego słowa.

**ASR adres efektywny**

Przesunięcie zadanego adresu efektywnego o jeden bit w prawo. Operacja wykonuje się na słowie. Dopuszczalnymi adresami efektywnymi są: (An), (An)+, -(An), x(An), x(An,R.s) oraz komórka pamięci.

Przykłady:

```
ASR.B   #1,D2
ASR.L   D7,D0
ASR     $60000
```

**Znaczniki:**

| | |
|---|---|
| X i C | - ustawiane zgodnie z ostatnim wysuniętym bitem |
| Z | - ustawiany gdy wynik jest równy zero |
| N | - ustawiany zgodnie z najstarszym bitem wyniku |
| V | - zawsze zerowany |

- **LSL - Logical Shift Left - przesunięcie logiczne w lewo.**

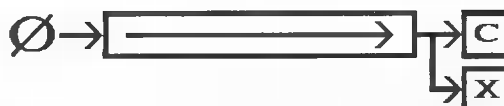
Przesunięcie logiczne w lewo jest to przesunięcie operandu o zadaną ilość bitów, przy czym najstarszy bit idzie do znaczników X i C, a do najmłodszego bitu wpisywane jest zero.

**LSL #liczba,Dx**

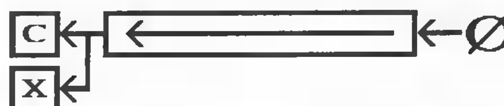
Przesunięcie rejestru danych o liczbę z zakresu od 1 do 8. Przesunięcie może się odbywać w zakresie bajtu, słowa oraz długiego słowa.

**LSL Dx,Dy**

Instrukcja LSR




Instrukcje ASL,LSL




Instrukcja ASR



Przesunięcie rejestru danych Dy o liczbę zawartą w rejestrze danych Dx. Przesunięcie może się odbywać w zakresie bajtu, słowa i długiego słowa.

**LSL adres efektywny**

Przesunięcie zadanego adresu efektywnego o jeden bit w lewo. Operacja wykonuje się na słowie. Dopuszczalnymi adresami efektywnymi są: (An), (An)+, -(An), x(An), x(An,R.s) oraz komórka pamięci.

Przykłady:

```
LSL.L   #4,D3
LSL.W   D5,D2
LSL     (A0)
```

**Znaczniki:**

X i C - ustawiane zgodnie z ostatnim wysuniętym bitem
Z - ustawiany gdy wynik jest równy zero
N - ustawiany zgodnie z najstarszym bitem wyniku
V - zawsze zerowany

**• LSR - Logical Shift Right - przesunięcie logiczne w prawo.**

Przesunięcie logiczne w prawo jest to przesunięcie operandu o zadaną ilość bitów przy czym najmłodszy bit idzie do znaczników X i C, a do najstarszego wpisywane jest zero.

**LSR #liczba,Dx**

Przesunięcie rejestru danych o liczbę z zakresu od 1 do 8. Przesunięcie może się odbywać w zakresie bajtu, słowa oraz długiego słowa.

**LSR Dx,Dy**

Przesunięcie rejestru danych Dy o liczbę zawartą w rejestrze danych Dx. Przesunięcie może się odbywać w zakresie bajtu, słowa i długiego słowa.

**LSR adres efektywny**

Przesunięcie zadanego adresu efektywnego o jeden bit w prawo. Operacja wykonuje się na słowie. Dopuszczalnymi adresami efektywnymi są: (An), (An)+, -(An), x(An), x(An,R.s) oraz komórka pamięci.

Przykłady:

```
LSR.B   #4,D4
LSR.L   D7,D0
LSR     (A0)+
```

**Znaczniki:**

X i C - ustawiane zgodnie z ostatnim wysuniętym bitem
Z - ustawiany gdy wynik jest równy zero
N - ustawiany zgodnie z najstarszym bitem wyniku
V - zawsze zerowany

I to są wszystkie instrukcje przesuwania, a teraz porcja kolejnych instrukcji.

**MOVEM - Move Multiple Registers - przesyłanie rejestrów.**

Instrukcja ta służy do szybkiego przesłania rejestrów do i z pamięci, i może się to odbywać na słowie bądź długim słowie a dozwolonymi trybami adresowania są:

**MOVEM rejestry,adres efektywny - zapisanie rejestrów.**

**MOVEM adres efektywny,rejestry - pobranie rejestrów.**

Instrukcji tej najczęściej używa się aby zapisywać i pobierać wartości rejestrów ze stosu na początku i końcu procedur. Przy zapisywaniu wartości rejestrów są odkładane w następującej kolejności: najpierw rejestry danych (od zerowego do siódmego), a potem rejestry adresowe (też w kolejności rosnącej). Instrukcja ta, jako bardzo szybka, może być używana do wypełniania obszarów pamięci pewnymi wartościami (na przykład szybkie czyszczenie danego obszaru).

AMIGA

A teraz kilka słów o składnii i przykłady. Listę rejestrów podajemy oddzielając je od siebie znakiem „/" (slash), natomiast blok rejestrów zapisujemy podając pierwszy i ostatni rejestr oddzielone znakiem „-" (minus), na przykład: D0/D1/D2/D3/D4/D5/D6/D7/A0/A1/A2/A3/A4/A5/A6/A7 (brrrrr....) możemy zastąpić krótszą kombinacją: D0-D7/A0-A7 (już lepiej).

Należy tylko pamiętać, że odsyłając rejestry na stos nie podajemy nigdy rejestru A7 gdyż to on jest wskaźnikiem stosu.

Przykład:

MOVEM.L D0-D7/A0-A6,-(SP) - odesłanie rejestrów na stos. Wskaźnik stosu zostanie zmniejszony o 60.
MOVEM.L (SP)+,D0-D7/A0-A6 - pobranie rejestrów ze stosu. Wskaźnik stosu zostanie powiększony o 60.
MOVEM.W D0-D3/D7/A4-A6,$60000 - odesłanie rejestrów D0,D1,D2,D3,D7,A4,A5,A6 pod adres $60000.

**Znaczniki:**

Żaden nie ulega zmianie.

**• MOVE to CCR - Move To Condition Codes Register - przesłanie danej do rejestru znaczników.**

**MOVE adres efektywny, CCR.**

Instrukcja powoduje przeniesienie słowa spod adresu wskazywanego przez adres efektywny do rejestru znaczników (warunków). Starszy bajt tego słowa jest ignorowany, natomiast na podstwie bitów młodszego słowa ustawiane są odpowiednie

**Znaczniki:**

X - ustawiany na podstawie bitu 4
N - ustawiany na podstawie bitu 3
Z - ustawiany na podstawie bitu 2
V - ustawiany na podstawie bitu 1
C - ustawiany na podstawie bitu 0

Dozwolone są wszystkie tryby adresowania za wyjątkiem adresowania bezpośredniego rejestru adresowego.

Przykład:

MOVE D0,CCR - ustawienie znaczników na podstawie bitów w rejestrze danych D0.

**• MOVE to SR - Move To Status Register - przesłanie do rejestru statusowego.**

**MOVE adres efektywny, SR.**

Instrukcja ta przesyła całe słowo do rejestru statusowego i na jego podstawie zmienia wszystkie bity rejestru statusowego. Wymagane jest jednak, aby przed wykonaniem tej instrukcji procesor był w trybie nadzorcy, to znaczy musi być ustawiony znacznik S (Supervisor - bit 13 w rejestrze statusowym SR). Instrukcja ta oczywiście modyfikuje bajty w rejestrze warunków, jednak do

AMIGA

modyfikacji znaczników warunków lepiej jest używać instrukcji MOVE to CCR, która może być wykonywana zarówno w trybie nadzorcy, jak i użytkownika.

Przykład:
MOVE $60000,SR - przesłanie słowa spod adresu $60000 do rejestru statusowego.

- **MOVE from SR - Move From Status Register - przesłanie z rejestru statusowego.**

**MOVE SR,adres efektywny**

Instrukcja ta przesyła zawartość rejestru statusowego pod adres wskazywany przez adres efektywny. Instrukcja ta nie jest instrukcją uprzywilejowaną, więc nie ma potrzeby ustawiania znacznika S (przejście w stan nadzorcy).

Przykład:
MOVE SR,-(A7) - przesłanie rejestru statusowego na stos.

**Znaczniki:**
Żaden ze znaczników nie ulega zmianie.

- **MOVE USP - Move User Stack Pointer - przesłanie wskaźnika stosu użytkownika.**

**MOVE USP,An**

**MOVE An,USP**

Instrukcja ma za zadanie przesłanie wskaźnika stosu użytkownika do rejestru adresowego lub zawartości rejestru adresowego An (n=0,1,...,7) do wskaźnika stosu użytkownika. Jest instrukcją uprzywilejowaną, a więc wymaga wprowadzenia procesora w stan nadzorcy, czyli należy ustawić bit S (Supervisor) w rejestrze statusowym SR.

Przykład:
MOVE USP,A0 - przesłanie zawartości wskaźnika stosu użytkownika do rejestru A0.
MOVE A5,USP - przesłanie zawartości rejestru A5 do wskaźnika stosu użytkownika (USP).

**Znaczniki:**
Żaden ze znaczników nie jest modyfikowany.

- **LINK - Link And Allocate - łączenie i alokacja.**

**LINK An, #ilość**

Instrukcja ta rezerwuje pewien roboczy obszar pamięci na stosie. Często używa sie jej w podprogramach, a zwłaszcza korzystają z niej kompilatory języków wysokiego poziomu. Dzięki tej instrukcji możemy otrzymać potrzebny obszar na zmienne lokalne wykorzystywane tylko jednorazowo przez dane procedury. Aby otrzymać ten obszar musimy jako parametry podać rejestr adresowy, który będzie nam wskazywał obszar zaalokowany oraz ilość komórek, które chcemy otrzymać. Ilość tych komórek jest zawsze wartością ujemną gdyż instrukcja działa w ten sposób, że najpierw na stosie odkładana jest zawartość rejestru adresowego, następnie wartość wskaźnika stosu zostaje przepisana do tego rejestru adresowego i do wskaźnika stosu zostaje dodana ilość bajtów jaką chcemy otrzymać. Do tak otrzymanych zmiennych lokalnych odwołujemy się oczywiście w trybie adresowania pośredniego względem rejestru adresowego, który wskazuje nam obszar otrzymany, z przesunięciem (oczywiście ujemnym).

Przykład:
LINK A5,#-20 - otrzymamy 20 bajtów na zmienne lokalne wskazywane przez rejestr A5.

**Znaczniki:**
Żaden ze znaczników nie jest modyfikowany.

- **UNLK - UnLink - zwolnienie obszaru zarezerwowanego.**

**UNLK An**

Instrukcja ta zwalnia wcześniej zaalokowany obszar na stosie za pomocą instrukcji LINK.

Przykład:
UNLK A5

**Znaczniki:**
Żaden ze znaczników nie jest modyfikowany.

- **TAS - Test And Set An Operand - testowanie i ustawienie operandu.**

**TAS adres efektywny**

Instrukcja TAS testuje określony adresem (wskazywanym przez adres efektywny) bajt. Najstarszy bit tego bajtu zostaje zawsze ustawiony, natomiast znaczniki N i Z zostają ustawione zgodnie z zawartością bajtu przed wykonaniem operacji.

Przykład:
TAS D0 - test bajtu w rejestrze danych D0 i ustawienie bitu 7 w tym rejestrze.

**Znaczniki:**
X - nie zmieniany.
N - kopia najstarszego bitu operandu przed wykonaniem operacji ustawienia tego bitu.
Z - ustawiany gdy przed operacją wszystkie bity operandu były wyzerowane.

**Marcin „Duddle" Dudar**

# telegram

Wśród wszystkich, którzy zamówią komplet dysków **PUBLIC DOMAIN PACK** za rok 1991 na **AMIGĘ**

- gwiazdkowa cena 240.000 zł

**ROZLOSUJEMY** 10 programów D-Mon professional

Do każdego zamówionego zestawu dołączamy gwiazdkowy prezent /gratis/ DISK BOX na 10 dyskietek 3.5''

Wśród wszystkich, którzy zamówią komplet dysków **PUBLIC DOMAIN PACK** za rok 1991 na **C - 64**

- gwiazdkowa cena 200.000 zł

**ROZLOSUJEMY** 10 programów VOICETRACKER V4.0

Do każdego zamówionego zestawu dołączamy gwiazdkowy prezent /gratis/ DISK BOX na 10 dyskietek 5.25''

Wśród wszystkich, którzy zamówią komplet taśm **PUBLIC DOMAIN PACK** za rok 1991 na **C - 64**

- gwiazdkowa cena 100.000 zł

**ROZLOSUJEMY** 10 programów VOICETRACKER V4.0

Wszyscy, którzy zakupią **D-Mon professional** w okresie gwiazdkowym otrzymują dodatkowo /gratis/ **MOUSE PAD!**

*Nasza oferta gwiazdkowa ważna jest dla wpłat dokonanych do 15 lutego 1992 roku!*

AMIGA

Opis biblioteki ARP (Amiga DOS Replacement Project) rozpoczęliśmy w numerze 10/91. Część drugą zawierał numer 12/91. Dziś przedstawiamy kolejne procedury tej biblioteki.

# ARP LIBRARY cz.3

**EscapeString - zamiana elementów ze znakiem „Esc" na wartości hexadecymalne**
**NowaDługość = EscapeString („Ciąg")**
**D0 A0**

Funkcja EscapeString przeszukuje ciąg w poszukiwaniu znaków poprzedzonych znakiem Esc i zamienia je na odpowiednie dla nich wartości kodów ASCII.

Aktualnie funkcja rozpoznaje następujące znaki specjalne:

N - nowa linia
V - tabulator pionowy
T - tabulator poziomy
B - cofnięcie w tył (backspace)
F - koniec strony (formfeed)
E - escape (znak o kodzie ascii 27)
Xnn - wartość reprezentowana przez szesnastkową wartość nn (np. /X7C)

Powyższe znaki mogą być jednocześnie podawane jako małe i duże. Jeżeli funkcja ta znajdzie poprzedzony znakiem esc znak nie znajdujący się w liście powyżej, to po prostu go skopiuje. I tak na przykład „/A" zostanie zastąpiony znakiem „A", a sekwencja „//" pojedynczym znakiem „/". Przy wprowadzaniu wartości szesnastkowych, argument „/X" może zawierać jeden bądź dwa znaki składające się na liczbę w kodzie szesnastkowym.

*Wejście:*

Ciąg - ciąg znaków zawierający kody poprzedzone kodem esc „/". Ciąg musi być zakończony zerem.

*Wyjście:*

NowaDługość - długość ciągu po zamianie wszystkich kodów.

**GetTracker - otrzymanie struktury śledzenia**
**Tracker = GetTracker()**
**A1**

Tworzy strukturę DefaultTracker, którą musimy zainicjować przypisując jej konkretne źródło (źródłami są biblioteki, urządzenia, itp.)

*Wejście:*

Nic.

*Wyjście:*

W A1 otrzymujemy adres struktury.

**GetAccess - dołączenie użytkownika dla danego źródła.**
**Wskaźnik = GetAccess (Tracker)**
**D0 a1**

Procedura zwraca wskaźnik do śledzonego źródła, jeżeli jest ono ciągle obecne. Jeżeli źródło zostało zwolnione przez FreeAccess to wskaźnik będzie wyzerowany.

*Wejście:*

Tracker - wskaźnik struktury, zawierający źródło do którego chcemy otrzymać dostęp.

*Wyjście:*

Wskaźnik - wskaźnik do źródła.

**FreeAccess - zmniejsza licznik użycia śledzonego źródła**
**FreeAccess (Tracker)**
**A1**

Zmniejsza licznik użytkowników danego źródła, w przypadku gdy licznik osiągnie -1, źródło zostanie automatycznie usunięte.

*Wejście:*

Tracker - wskaźnik struktury tracker dla danego źródła.

**LDiv - dzielenie 32 bitowe**
**Wynik = LDiv (Dzielna, Dzielnik)**
**D0 D0 D1**

Assembler MC68000 nie posiada zaimplementowanego dzielenia dwóch 32 bitowych liczb. Ta procedura poprawia ten błąd.

*Wejście:*

Dzielna - 32 bitowa liczba
Dzielnik - 32 bitowa liczba.

*Wyjście:*

Wynik - 32 bitowy wynik dzielenia dzielnej przez dzielnik.

**LMod - reszta dzielenia 32 bitowego**
**Wynik = LMod (Dzielna, Dzielnik)**
**D0 D0 D1**

W wyniku działania tej procedury otrzymujemy resztę z dzielenia jednej 32 bitowej liczby przez drugą. Wynik posiada taki sam znak jaki posiadała dzielna.

*Wejście:*

Dzielna - 32 bitowa liczba
Dzielnik - 32 bitowa liczba.

*Wyjście:*

Wynik - 32 bitowa reszta dzielenia dzielnej przez dzielnik ze znakiem dzielnej.

**LMult** - **mnożenie 32 bitowe**
**Wynik** = **LMult (Wartość1, Wartość2)**
**D0** D0 D1

Assembler MC68000 nie posiada zaimplementowanego mnożenia liczb 32 bitowych. Ta funkcja robi to zastępczo.

*Wejście:*

Wartość1 - 32 bitowa liczba
Wartość1 - 32 bitowa liczba.

*Wyjście:*

Wynik - 32 bitowy wynik mnożenia dwóch liczb.

**PatternMatch** - **porównanie ciągu ze wzorem**
**Wynik** = **PatternMatch ( Wzór , Ciąg )**
**D0** A0 A1

Funkcja ta porównuje ciąg z podanym wzorem i określa czy zachodzi właściwa kombinacja. Gdy tak, to w wyniku zwraca wartość nieujemną. Można używać tej funkcji do określania zgodności tekstu np. w komendach wyszukiwania (Search) czy w FileRequesterach.

Wzór musi być złożony ze specjalnych elementów:

(p1|p2|p3) -porównanie ciągu z jednym z ciągów p1, p2, p3.
? -dowolny pojedynczy znak
#ąg -dany ciąg powtórzony 0 lub więcej razy
* -0 lub więcej powtórzeń dowolnego znaku

*Wejście:*

Wzór - ciąg znaków mogący zawierać elementy podane powyżej, np. *.* - odszukanie ciągu zawierającego kropkę ('.'). Jednak ciąg ten przed podaniem musi być przetworzony procedurą PreParse.
Ciąg - ciąg znaków porównywany ze wzorem.

*Wyjście:*

Wynik - wartość dodatnia oznacza sukces.

**PreParse** - **stworzenie ciągu elementarnego**
**Wynik** = **PreParse (CiągWej, CiągWyj)**
**D0** A0 A1

Procedura tworzy z ciągu zawierającego WildCards'y ciąg dla procedury PatternMatch.

*Wejście:*

CiągWej - ciąg zawierający wildcards'y np. *, ?, itd.
CiągWyj - wskaźnik bufora gdzie będzie stworzony ciąg dla PatternMatch.
Długość bufora nie musi być większa od długości ciągu wejściowego.

*Wyjście:*

Wynik - prawda w przypadku, gdy nastąpiła konwersja.

**Marcin „Duddle" Dudar**




## Pacland

Na tytułowym obrazku wpisz „AVALON", jeśli ekran błyśnie to uzyskałeś nieśmiertelność. Jeśli to nie poskutkuje - spróbuj inaczej. Rozpocznij grę dla 2 graczy. Dojdź pierwszym jak najdalej. Drugim musisz dojść do wróżki, która da ci buty. W drodze powrotnej pchnij trzeci kaktus, aż ukaże się żółty pacman. Gdy go weźmiesz natychmiast się zabij. Graj dalej pierwszym graczem. Jeśli zginiesz, to powtórz operację drugim graczem. Jak długo gracz drugi bierze żółtego pacmana, tak długo gracz pierwszy jest nieśmiertelny.

## Pipeline

Oto kody do gry: FOLD, EYES, EGGS, TEAR, PEAS, DUCT, PODS.

## PipeMania

Oto niektóre hasła do gry: GRIP, TICK, DUCK, OOZE, BLOB, BALL, WILD.

## R - Type

Wpisz „SUMITA" w tabelę wyników, by uzyskać nieśmiertelność.

## Rambo III

Wpisz się na listę wyników jako „RENEGADE". Od tej pory wciskając 1, 2 lub 3 przenosisz się do kolejnych stref.

## Return of the Jedi

Wpisz się na tabelę wyników jako „Darth Vader". Teraz wciskając F2 zmieniasz strefy.

## Rick Dangerous

Wpisz „POOKY" w tabelę wyników, aby kontynuować grę w strefie, gdzie zginąłeś.

**Robert „Mr.Raf" Turliński**

AMIGA

W poprzednim odcinku mówiliśmy o przekształceniach typów, deklarowaniu i definiowaniu funkcji, przekazywaniu parametrów funkcji (przez wartość) oraz zasięgu zmiennych deklarowanych wewnątrz funkcji.

# KURS JĘZYKA C (cz.4)

Zajmijmy się nieco szerzej zasięgiem zmiennych. Pamiętamy z poprzedniej części kursu, że jeżeli wewnątrz jakiejś funkcji zadeklarowaliśmy zmienną, to zmienna ta istniała tylko wewnątrz danej funkcji, np.

```
void Liczba()
{
 int a;
 a = 5;
 printf("\nZmienna a ma wartosc %d\n", a );
}

main()
{
 Liczba();
}
```

W powyższym przykładzie widzimy, że w funkcji Liczba() została zadeklarowana zmienna, której wartość ustalono na równą 5. Zmienna ta istnieje tylko wewnątrz funkcji Liczba(). Nic nowego. Jeżeli jednak będziemy wielokrotnie wywoływać w naszych programach jakąś funkcję to należy pamiętać, że sposób zadeklarowania zmiennej, jak w powyższym przykładzie powoduje, że zmienna przy kolejnych wywołaniach może mieć wartość przypadkową.

Standard języka C wyróżnia cztery typy klas pamięci: **auto, extern, static i register.** Zwykle, jeśli nie zażyczymy sobie inaczej, zmienne deklarowane w funkcjach należą do klasy **auto**. Oznacza to, że w momencie wywoływania funkcji zostaje przydzielona pamięć dla zmiennych w tej funkcji występujących. Ponieważ zwykle jest to pierwsze lepsze z brzegu miejsce, przydzielenie pamięci np. zmiennej typu char wyglada mniej więcej tak: „Hej, ty, zmienna char! Zaczynasz się od adresu xxxxxx!". Co jest pod tym adresem, to nikogo nie interesuje. Wynika stąd pierwsza uwaga: zmienne **auto** mają początkowe wartości przypadkowe. Po zakończeniu działania funkcji zmienna **auto** jest usuwana z pamięci, co z kolei tłumaczy fakt, że przy ponownym wywołaniu funkcji wartość tej zmiennej jest ponownie przypadkowa. W obrębie funkcji możemy określoną daną zadeklarować jako typ auto, poprzedzając deklaracje słowem auto, np.:

```
auto int test;
```

W praktyce się tego nie stosuje, gdyż - jak wspomniałem - każda zmienna bez określonej klasy pamięci jest traktowana jako **auto**.

Druga klasa pamięci to **static**. Od auto różni się tym, że zmienne static są inicjowane zerami oraz tym, że po zakończeniu działania funkcji obszar pamięci zajmowany przez zmienną typu static nie jest zwalniany. Konsekwencja tego faktu jest prosta: zmienna typu **static** zachowuje swoją wartość pomiędzy poszczególnymi wywołaniami! Oto przykład, który różni się od wcześniejszego jedynie deklaracją zmiennych:

```
void Liczba()
{
 static int a;
 static int b;

 a = 5;
 printf("\nZmienna a ma wartosc %d", a );
 printf("\nZmienna b ma wartosc %d\n", b ),
 b = 6;
}

main()
{
 Liczba();
 Liczba();
}
```

Trzecia klasa pamięci jest stosunkowo najprostsza do przyswojenia. Tą klasą jest typ **extern**. Zmienne typu **extern** deklaruje się poza wszystkimi funkcjami (także poza main() ).

```
int a;

main()
{
 printf( "\nZmienna zewnetrzna a jest rowna %d\n", a );
}
```

Zmienne typu **extern** istnieją przez cały czas działania programu i są widziane ze wszystkich funkcji. Zmienne tego typu przy inicjalizowaniu otrzymują wartość zero, chyba że nadajemy im od razu inną wartość. Zbyt duża ilość zmiennych typu **extern** zaciemnia obraz programu, trudniej jest uchwycić pewne powiązania istniejące

pomiędzy zmiennymi czy funkcjami. Z tego powodu należy starać się unikać nadużywania zmiennych **extern**, choć nie oznacza to, że należy z nich rezygnować. Dzięki temu, że zmienne te są widziane z każdej funkcji, mogą być wykorzystywane (i z reguły są) do przenoszenia danych pomiędzy funkcjami. Pamiętajmy jednak, że w dużym programie łatwo stracić kontrolę nad tym, co modyfikuje określona zmienna extern.

Ostatną klasą pamięci jest **register**. Zmienne tego typu przybierają przy inicjowaniu wartości przypadkowe. Cały zaś „bajer" tej klasy pamięci polega na tym, że kompilator w miarę możliwości stara się upchnąć te zmienne w rejestrach procesora, dzięki czemu można uzyskać znaczne zwiększenie prędkości działania programu.

Zajmijmy się teraz zwracaniem przez funkcje różnych wartości. Jak wspomniałem, zmienne typu **extern** można wykorzystać do przenoszenia danych pomiędzy funkcjami. Oto przykład:

```
int    Zmienna_zewnetrzna;

void Wartosc()
{
 Zmienna_zewnetrzna = 3;
}

void Druk()
{
 printf( "\nZmienna zewnetrzna jest rowna %d\n",
 Zmienna_zewnetrzna  );
}

main()
{
 Wartosc();
 Druk();
}
```

Pojawił się jednak pewien problem. Jak zmodyfikować zmienne innego typu niż **extern**? Nie można przecież wszystkich danych deklarować jako zewnętrzne. Poza tym, co zrobić, gdy fukcja ma zwrócić więcej niż jedną wartość? Odpowiedzi na te pytania są dwie: można wykorzystać **wskaźniki** lub **struktury**. Temat „struktury" rozwiniemy w następnej części cyklu, teraz zajmijmy się wskaźnikami.

**Wskaźnik** nie jest żadną straszną bestią, aczkolwiek korzystając z jego pomocy trzeba uważać, co się robi. Każda zmienna ma przydzielony pewien adres pamięci. Zmiana wartości zmiennej następuje poprzez zmianę tego, na co ten adres wskazuje. Zmienne mogą być różnej wielkości, np.: w przypadku zmiennej o rozmiarze bajta, modyfikacja jej wartości polega ma zmianie bajta znajdującego się pod przydzielonym jej adresem. Pisząc program nie zastanawiamy się nad tą operacją, załatwia to za nas kompilator, który wie, że zmienna o takiej i takiej nazwie znajduje się tam i tam. Czasem jednak warto, by się tą wiedzą z nami podzielił.

Jeżeli funkcji przekażemy nie wartość zmiennej (poprzez podanie nazwy tej zmiennej), lecz wskaźnik, wtedy mamy bezpośredni dostęp do tej danej! Jak to się odbywa? Zobaczmy to na najbardziej klasycznym przykładzie funkcji, która zamienia wartości dwóch danych.
W poniższym programie zamiana nie nastąpi, bo nie korzystalismy ze wskaźników, lecz przekazaliśmy funkcji swap() argumenty przez wartość:

```
/* Z tego nic dobrego
nie wyjdzie */

void swap ( int a, int b )
{
 int c;

 c = a;
 a = b;
 b = c;
}

main()
{
 int a, b;

 a = 3;
 b = 6;

 printf("\nZmienna a ma wartosc %d, a zmienna b jest
 rowna %d\n", a, b );
 swap( a, b ); printf("\nZmienna a ma wartosc %d, a
 zmienna b jest rowna %d\n", a, b );
}
```

Poniższy przykład będzie już prawidłowy:

```
void swap ( int *a, int *b )
{
 int c;

 c = *a;
 *a = *b;
 *b = c;
}

main()
{
 int a, b;

 a = 3;
 b = 6;

 printf("\nZmienna a ma wartosc %d, a zmienna b jest
 rowna %d\n", a, b );
 swap( &a, &b );
 printf("\nZmienna a ma wartosc %d, a zmienna b jest
 rowna %d\n", a, b );
}
```

**Wskaźnik** deklaruje się pisząc przed jego nazwą znak *. W powyższym programie argumenty funkcji swap() zostały zadeklarowane jako wskaźniki do zmiennych typu int:

```
int *a;
int *b;
```

UWAGA! W deklaracji znak * jest tylko informacją, że deklarowana dana jest wskaźnikiem.

AMIGA

Oto inne przykłady:

```
char *wskaznik;
long int *wskaznik;
float *wskaznik;
```

Taka deklaracja mówi, że zmienna wskaźnik zawiera adres do zmiennej określonego typu (np. char czy float).

Jak jednak znaleźć adres określonej zmiennej? Sprawę załatwia operator adresu &. Jeżeli przed nazwą danej napiszemy &, np. (jak w przykładzie): &a, to oznacza: „adres zmiennej a". Teraz już widać, co się działo w programie. Funkcji swap() przekazaliśmy adresy dwóch zmiennych: a oraz b ( swap( &a, &b ) ). Wszystko się zgadza, bo argumentami swap() są właśnie wskaźniki
( swap ( int *a, int *b ) ).

No dobrze, ale my mamy przecież ADRESY danych, a chcemy zmienić ich WARTOŚĆ! Tu problem rozwiązuje znany już operator *. Jeśli postawimy go przed wskaźnikiem (np. *a ) to wyrażenie to oznacza zawartość danej wskazywanej przez wskaźnik a. Tak więc nasza funkcja swap() na dobrą sprawę zwraca więcej niż jedna wartość, bowiem w wyniku jej działania uległy modyfikacji dwie podane zmienne.

### PAMIETAJMY:

1. W deklaracji znak * czytamy: „deklarowany jest wskaźnik do typu...", np.:
   int *a;
   czytamy: „wskaźnik do zmiennej a, która jest typu int".
2. W wyrażeniu znak * czytamy: „wartość zmiennej wskazywanej przez...", np.:
   *a = 5;
   czytamy: „wartość zmiennej wskazywanej przez a jest równa 5".
3. Operator & czytamy jako „podaj adres zmiennej...", np.:
   test( &a );
   czytamy: „przekaz funkcji test() adres (wskaźnik) do zmiennej a".

Na zakończenie tej części kursu jeszcze kilka uwag o wskaźnikach. Otóż ze wskaźnikami mieliśmy już doczynienia wcześniej. Bawiliśmy się już przecież tablicami! Jak taka tablica wygląda w pamięci komputera?
W momencie deklaracji jest rezerwowany pewien obszar pamięci. Nazwa tablicy to nic innego jak adres początku tego obszaru! Kompilator wyszukuje poszczególne elementy tablicy w ten sposób, że wiedząc, do którego elementu się odwołujemy, wykonuje operacje:

```
adres bazowy + numer elementu * rozmiar elementu.
```

Znajdźmy adres początku jakiejś tablicy:

```
main()
{
 int a[10];
 printf( "\nTablica zaczyna sie od adresu %d\n", a );
}
```

Innymi słowy adres pierwszego elementu tablicy z naszego przykładu można uzyskać pisząc: &a[0] lub po prostu a.

UWAGA! Nazwa tablicy jest stała, nie można zmieniać jej wartości, natomiast wskaźniki są zmiennymi!

Jeśli mamy zadeklarowany wskaźnik do zmiennej typu int:

```
int *a;
```

to wyrażenie *a oznacza wartość zmiennej typu int. Natomiast wyrażenie

```
*(a+1)
```

oznacza po prostu kolejny element typu int. Postępując według wzoru:

```
*(adres + przesuniecie)
```

odwołujemy się do kolejnego elementu wskazywanego przez wskaźnik. Przesunięcie nastąpi zawsze o rozmiar wskazywanej danej. Z tego płynie następujący wniosek:

element tablicy np. a[2] może zostać zapisany jako *(a+2). Adres do elementu, np. &a[5] może być zapisany jako (a+5).

Kolejna część kursu poświęcona będzie strukturom.

**Jarosław Chrostowski**

Zapraszamy wszystkich do udziału w stałym konkursie pod hasłem:

## Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace. Tematyka programów dowolna. Konkurs rozgrywany jest osobno dla komputerów C-64 i Amiga.
Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane).
Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu.
Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po **500.000** zł każda.
Decyzje Sądu Konkursowego są nieodwołalne.
Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma.
Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

**Redakcja**

O blitterze już wspominaliśmy na łamach naszego pisma (patrz: nr 7,8/1991, artykuł: O blitterze słów kilka), i jak wszyscy dobrze wiedzą jest to bardzo kochane „urządzonko" w naszej Amisi, które potrafi dokonywać operacji logicznych na pamięci, potrafi ją wypełniać, a na dodatek może kreślić kreseczki.

# NARYSOWAĆ KRESKĘ
## czyli znowu małe co nieco o blitterze

I dzisiaj zajmiemy się właśnie tymi kreseczkami. Aby rozpocząć pracę nad kreśleniem musimy poznać inne znaczenia komórek BLTCON0 i BLTCON1.

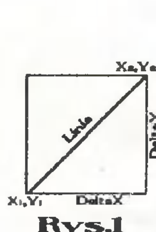
**BLTCON0**

| Bit | Nazwa | Funkcja |
|---|---|---|
| 15 | START3 | Cztery najmłodsze bity pozycji startowej X |
| 14 | START2 | |
| 13 | START1 | |
| 12 | START0 | |
| 11 | USEA = 1 | Rejestry USE A,B,C,D ustawiamy |
| 10 | USEB = 0 | na używanie A, C i D |
| 9 | USEC = 1 | |
| 8 | USED = 1 | |
| 7-0 | LF7 - LF0 | Ustawiamy na kombinacje D = aC+AB czyli wartość $CA |

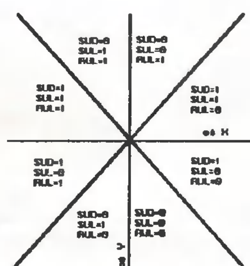
**BLTCON1**

| Bit | Nazwa | Funkcja |
|---|---|---|
| 15 | TXT3 | Maska, która powinna zaczynać się |
| 14 | TXT2 | od pierwszego bitu w słowie |
| 13 | TXT1 | |
| 12 | TXT0 | |
| 11-7 | 0 | Nie używane |
| 6 | SIGN | Ustawiony gdy 2*DeltaM<DeltaW |
| 5 | 0 | Nie używany |
| 4 | SUL | Bity ustawiane oktantem |
| 3 | SUD | czyli kierunkiem kreślenia linii. |
| 2 | AUL | |
| 1 | SING | Ustawiamy, gdy kreślimy linie do wypełniania |
| 0 | LINE = 1 | Zawsze ustawiony (kreślenie linii). |

Teraz gdy znamy już przeznaczenie rejestrów, możemy zająć się teorią kreślenia linii. Naszą przykładową linię będziemy kreślić od punktu określonego współrzędnymi X1,Y1 do punktu określonego współrzędnymi X2,Y2 (rys.1). Najpierw musimy wyznaczyć kierunek kreślenia linii (rys.2) czyli tak zwany oktant, oraz zdefiniować dwie ważne wielkości, a mianowicie różnicę pomiędzy współrzędnymi X1 i X2 oraz Y1 i Y2 od tej chwili nazywane DeltaX i DeltaY gdzie:



Rys.1


Rys.2

**DeltaX = |X1-X2|**
**DeltaY = |Y1-Y2|**

Bity 4, 3 i 2 w komórce BLTCON1 (SUL, SUD, AUL) to właśnie oktant jaki otrzymujemy z wykresu.

Możemy także wyliczyć je z prostej zależności:

| Współrzędne | | | Bity 432 |
|---|---|---|---|
| Y1 | <= | Y2 | 1 1 0 |
| X1 | <= | X2 | |
| DX | >= | DY | |
| Y1 | <= | Y2 | 0 0 1 |
| X1 | <= | X2 | |
| DX | <= | DY | |
| Y1 | <= | Y2 | 0 1 1 |
| X1 | >= | X2 | |
| DX | <= | DY | |
| Y1 | <= | Y2 | 1 1 1 |
| X1 | >= | X2 | |
| DX | >= | DY | |
| Y1 | >= | Y2 | 1 0 1 |
| X1 | >= | X2 | |
| DX | >= | DY | |
| Y1 | >= | Y2 | 0 1 0 |
| X1 | >= | X2 | |
| DX | <= | DY | |
| Y1 | >= | Y2 | 0 0 0 |
| X1 | <= | X2 | |
| DX | <= | DY | |
| Y1 | >= | Y2 | 1 0 0 |
| X1 | <= | X2 | |
| DX | <= | DY | |

Teraz gdy mamy już wartości bitów SUD, SUL, AUL musimy wyznaczyć jeszcze dwie wartości, a mianowicie DeltaM i DeltaW, gdzie:

**DeltaM = min(DeltaX,DeltaY)**
**DeltaW = max(DeltaX,DeltaY)**

Jeżeli wartość 2*DeltaM jest mniejsza od DeltaW (2*DeltaMaW) to musimy ustawić bit SIGN w komórce BLTCON1, natomiast w przeciwnym wypadku musimy ten bit wyzerować. Teraz kolejno ustawiamy komórki blittera:

BLTAPTL - tutaj wpisujemy wartość równą 2*DeltaM-DeltaW

BLTCPT i BLTDPT - to komórki zawierające adres pierwszego słowa, w którym będzie kreślona linia a ustalany ze wzoru:

Adres = Adres Ekranu + (Wysokość-Y1-1)*Szerokość + 2*(X1/16) BLTAMOD - ta komórka musi zawierać wartość 2*DeltaM-2*DeltaW

BLTBMOD - tutaj wpisujemy 2*DeltaM

BLTCMOD i BLTDMOD - tutaj wpisujemy szerokość ekranu, na którym kreślimy (oczywiście w bajtach).

BLTADAT - do tej komórki musimy wpisać wartość $8000.

BLTBDAT - tutaj wpisujemy maskę dla linii (standardowo $ffff).

BLTAFWM - rejestr maski musi zawierać $ffff.

AMIGA

BLTSIZE - tutaj powinno się wpisać długość linii do bitów od 6 do 15 oraz 2 do bitów od 0 do 5. Długość linii to wielkość zawarta w DeltaW. Przykład:
Jeżeli chcemy wykreślić linię z punktu X1=10, Y1=190 do punktu X2=100, Y2=25 na ekranie o adresie $70000 i szerokości 320 punktów (40 = $26 bajtów) oraz wysokości 256 linii to musimy wykonać następujące czynności:

**1) Obliczenie wielkości: DeltaX, DeltaY, DeltaM, DeltaW**
DeltaX = |10-100|=|-90|=90
DeltaY = |100-25|=|75|=75
DeltM=min(75,9)=75
DeltW=max(75,)=90

**2) Ustawienie bitów SUD, SUL, AUL,SIGN**
DeltaXDeltaY, X12, Y1Y2 czyli bity bądą ustawione następującymi wartościami:
SUL = 1
SUD = 0
AUL = 0
SIGN = 0 bo 2*DeltaMDeltaW

**3) Ustalenie adresu kreślenia:**
Adres = $70000+(256-190-1)*40+2*(10/16)=$70a28
Wartość tę wpisujemy do komórek BLTCPT i BLTDPT a szerokość ($28) wpisujemy do komórek BLTCMOD i BLTDMOD.

**4) Ustalenie bitów START3-0 (komórka BLTCON0):**
START3-0 = X1 AND $F = 10 AND $F = $a
Czyli bit START3 = 1, START2 = 0, START1 = 1, START0 = 0

**5) Ustalenie wartości BLTAPTL, BLTAMOD, BLTBMOD:**
BLTAPTL =2*DeltaM-DeltaW = 150-90 = 60
BLTAMOD =2*DeltaM-2*DeltaW = 150-180 = -30
BLTBMOD =2*DeltaM = 150

**6) Ustalenie wartości komórki BLTSIZE:**
BLTSIZE = DeltaW*64+2 = 90*64+2 = $1682

**7) Ustalenie wartości BLTCON0 i BLTCON1:**
BLTCON0 = START3-0<12+%1011<8+$CA = $ABCA
BLTCON1 = Ustawione tylko bity LINE i SUL = $0011

W języku maszynowym wykreślenie takiej linii wyglądało by następująco:

```
lea     $dff000,a6
move.l  #$70a28,   BLTCPTH(a6)
move.l  #$70a28,   BLTDPTH(a6)
move.w  #$28,      BLTCMOD(a6)
move.w  #$28,      BLTDMOD(a6)
move.w  #60,       BLTAPTL(a6)
move.w  #-30,      BLTAMOD(a6)
move.w  #150,      BLTBMOD(a6)
move.w  #$abca,    BLTCON0(a6)
move.w  #$0011,    BLTCON1(a6)
move.w  #$1682,    BLTSIZE(a6)
```

I na koniec, procedura kreślenia linii

```
;* DRAW LINE *
; d0-X1 , d1-Y1 , d3-X2 , d4-Y2 , a0 - adres ekranu
Draw:
        move.l  #40,d4          ;40 - szerokosc ekranu
        mulu    d1,d4           ;Y1*Szerokosc
        moveq   #-$10,d5        ;Bity ktore zostana
        and.w   d0,d5
        lsr.w   #3,d5           ;Slowo w ktorym
                                znajduje sie punkt startowy
        add.w   d5,d4
        add.l   a0,d4;Adres dla pierwszego
                                punktu
;Wyliczanie Oktantu
        moveq   #0,d5
        sub.w   d1,d3           ;Y2-Y1, DeltaY w D3
        roxl.b  #1,d5           ; Jezeli DeltaY to d5=1
        tst.w   d3
        bge.s   Y2gY1           ;Skok gdy DeltaY
                                jest dodatnia
        neg.w   d3              ;Otrzymanie DeltaY
                                dodatniej
Y2gY1:
        sub.w   d0,d2           ;X2-X1, DeltaX w D2
        roxl.b  #1,d5           ;Bit przeniesienia do D5
        tst.w   d2
        bge.s   X2gX1           ;Skok gdy Delta136ksza
                                od DeltaX
        exg     d2,d3           ;Zamiana DeltaY z DeltaX
dYgdX:
        roxl.b  #1,d5           ;Bit przeniesienia do
                                d5 (numer oktantu)
        move.b  oktant
                (pc,d5),d5      ;Wartosc oktantu do d5
        add.w   d2,d2           ;d2 = DeltaM*2
WaitB:
        btst    #14,
                dmaconr(a6)     ;odczekanie az blitter
                                skonczy
        bne.s   WaitB           ;poprzednie zadanie
        move.w  d2,
                bltbmod(a6)     ;DeltaM*2 do
                                BLTBMOD(a6)
        sub.w   d3,d2           ;d2 = DeltaM*2-DeltaW
        bge.s   SignNL          ;Czy ustawic bit SIGN
        or.b    #$40,d5         ;Ustawienie bitu SING
SignNL:
        move.w  d2,
                bltaptl(a6)     ;DeltaM*2-DeltaW
                                do BLTAPTL
        sub.w   d3,d2           ;d2 = DeltaM*2-DeltaW*2
        move.w  d2,
                bltamod(a6)     ;DeltaM*2-DeltaW*2
                                do BLTAMOD
        move.w  #$8000,bltadat(a6)
        move.w  #$ffff,bltbdat(a6)
        move.w  #$ffff,bltafwm(a6)
        and.w   #$f,d0          ;Najmlodsze bity z X1
        ror.w   #4,d0           ;Przesuniecie na poz 15-12
        or.w    #$bca,d0        ;Dodanie bitow USEx i LFx
        move.w  d0,bltcon0(a6)
        move.w  d5,
                bltcon1(a6)     ;Ustawienie
                                komorki BLTCON1
        move.l  d4,
                bltcpth(a6)     ;Adres pierwszego
                                punktu linii
        move.l  d4,bltdpth(a6)
        move.w  #40,
                bltcmod(a6)     ;40 - szerokosc ekranu
        move.w  #40,bltdmod(a6)
        lsl.w   #6,d3           ;DeltaW * 64
        addq.w  #2,d3           ;d3 = DeltaW*62 + 2
        move.w  d3,
                bltsize(a6)     ;Wpisanie d3 do BLTSIZE
                                i uruchominie blittera
        rts
Oktant:
        dc.b    0*4+1
        dc.b    4*4+1
        dc.b    2*4+1
        dc.b    5*4+1
        dc.b    1*4+1
        dc.b    6*4+1
        dc.b    3*4+1
        dc.b    7*4+1
```

I to by było wszystko jeżeli chodzi o kreślenie linii, zapraszam wszystkich do eksperymentowania z liniami...

**Marcin „Duddie" Dudar**

# GWARANCJĘ SYSTEMATYCZNEGO OTRZYMYWANIA NASZEGO PISMA ZAPEWNIA TYLKO

# PRENUMERATA

**Wśród wszystkich, którzy do 25 stycznia 1992 roku /decyduje data stempla pocztowego/ zamówią całoroczną prenumeratę**

# ROZLOSUJEMY

## AMIGĘ CDTV!
## 20 szt. joystick'ów
## oraz
## 100 upominków
## /o wartości ok. 20.000 zł każdy/




Od stycznia 1992 r. nasze pismo kosztować będzie 10.000 zł. Wszyscy którzy do 25 stycznia wykupią roczną prenumeratę mogą skorzystać ze zniżki wpłacając nie 120.000 zł, a tylko 110.000 zł.

Wpłaty należy przesyłać na konto: BANK PKO S.A. Bydgoszcz, konto nr 5.09011-400522.7-136-11-111.0.

Blankiety wpłat powinny być czytelnie wypełnione i zawierać następujące informacje: imię i nazwisko lub nazwę instytucji, dokładny adres zamawiającego, liczbę zamawianych egzemplarzy oraz okres prenumeraty.

**Gwiazdkowe upominki - szczegóły wewnątrz numeru**